U.S. DEPARTMENT OF THE INTERIOR
GEOLOGICAL SURVEY

Nonlinear least-squares inversion of transient soundings for a central induction loop system (Program NLSTCI)

by

Walter L. Anderson

Open-File Report 82-1129

1982

DISCLAIMER

This program was written in FORTRAN-77 for a VAX-11/780 (VMS version 2.5) system*. Although program tests have been made, no guarantee (expressed or implied) is made by the author regarding program correctness, accuracy, or proper execution on all computer systems.

----

* Any use of trade names in this report is for descriptive purposes only and does not imply endorsement by the U.S. Geological Survey. This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards.

CONTENTS

INTRODUCTION

The inversion of transient soundings for a central induction loop system on a layered halfspace is provided by program NLSTCI. The numerical technique uses a general adaptive nonlinear least-squares algorithm originally developed by Dennis and others (1979), and extended externally for constrained nonlinear regression by Anderson (1982). The corresponding forward problem solution--also required in the inverse solution--is defined in Anderson (1981). The numerical integrations used in NLSTCI are by adaptive digital linear filtering as described in Anderson (1975) and Anderson (1979). Because digital convolution (filtering) methods are used, practical solutions for layered earth models are reasonably fast on most computers.

This report summarizes the general nonlinear least-squares (NLS) method used in Anderson (1982), but as applied to observed transient soundings obtained using a central induction loop system placed on an assumed horizontally layered earth model. In addition, the quasi-static case is assumed (i.e., displacement currents are neglected). The system must use an "on-off" step current source of arbitrary current, where the transient decay voltage is measured during the off-time (i.e., after t>0 sec.). An arbitrary maximum of 10-layers (homogeneous and isotropic) may be used; however, with most present time-domain electromagnetic (TDEM) measurement systems, only a few layers are generally resolvable for the given time range.

To avoid repeating the notation and other details of the forward problem solution in this report, the reader is referred to Anderson (1981)--which has been updated from the original published version. Similarly, details on the NLS method may be found in Anderson (1982). The present report will provide a brief description of the calculations, specific program parameters, and the VAX operating instructions. Appendix 1 offers some suggestions in converting the VAX program to other computer systems; Appendix 2 lists a simple input/output test example (taken from a known forward solution model); and Appendix 3 gives a partial source listing (the complete source is available on the distributed tape, as described in Appendix 3).

SUMMARY OF CALCULATIONS

The NLS method described in Anderson (1982) requires a twice-continuously differentiable nonlinear objective function F describing the model equation as a function of the unknown layer parameters (i.e., the conductivities and thicknesses of an MM-layered earth, MM>0). In this case, F is given by the transient V(t) defined in Anderson (1981, p.5), as

$$V(t) = \frac{2}{\pi} C \int_0^\infty \mathrm{Re}[Hz(\sqrt{b})/DC] \cos(bT)\, db, \qquad (1)$$

where discrete observed values $[V(t_i),t_i,\ i=1,2,...,N]$ are given. In some cases (e.g., data stacking), an associated standard deviation s_i may also be known, and should be used for a weighted least-squares solution (see parameter IWT=1 in Anderson, 1982, p.14). Note that the constant C in eq.(1), and in Anderson (1981,p. 4), should be corrected to read: $C=(nA)I/[\sigma_1(a^2+z^2)^{3/2}]$--see Anderson (1981, p. 4-5) for definitions of all symbols used.

Optionally, F may be given in terms of converted apparent resistivity ρ_a (see $INIT parameter IOPT=1 below) instead of voltage V(t) by equating the layered earth transient response to a halfspace response. In this case, the observed transient data $[V(t_i),t_i]$ must be converted to apparent resistivity data $[\rho_a(t\ ),t_i]$ by solving the following nonlinear equation (Raab and Frischknecht, 1982) for x at each i,

$$3\ \mathrm{erf}(x)-(3x+2x^3)\frac{2}{\sqrt{\pi}}\exp(-x^2)-2x^2T_iV(t_i) = 0, \qquad (2)$$

where C=1 is assumed in eq.(1), $T_i=2t_i/(\mu_0\sigma_1 a^2)$ is normalized time, and the units are V(volts/amp), t(seconds), and ρ_a(ohm-meters). For each solution of root x at an observed t_i, the apparent resistivity is given by

$$\rho_a = (\mu_0 a^2)/(4t_i x^2). \qquad (3)$$

When F is defined as in eq. (1) and IOPT=0 (default), then any convenient unit may be used for V (e.g., volts/amp, millivolts/amp, etc.), since the constant C in eq. (1) can be determined in the least-squares to account for a scale (or amplitude shift) factor times V(t).

For either IOPT=0 or 1 cases, the independent time variable $t>0$ must be given in <u>seconds</u> and in ascending order, and is assumed to be known without error.

The unknown (nonlinear) model parameters are denoted by the vector B(J), and has the following assumed order:

B(1),B(2),...,B(MM) are the MM-layer conductivities (in mhos/m.),

B(MM+1),B(MM+2),...,B(2*MM-1) are the MM-1 layer thicknesses (in m.), and

B(2*MM) is a transient scaling (or amplitude shift) parameter depending on the form of F chosen via $INIT parameter IOPT.

Thus, the discrete objective function F may be expressed for either IOPT=0 as

$$\left.\begin{array}{l} F = B(2*MM)\ [V(t_i, B(J),\ J=1,2,\ldots,2*MM-1)/B(1)], \\ \text{or for IOPT=1 as} \\ F = B(2*MM)\ [\rho_a(t_i, B(J),\ J=1,2,\ldots,2*MM-1)], \end{array}\right\} \quad (4)$$

where $i=1,2,\ldots,N$ and $N>2*MM\geq 2$ $(1\leq MM\leq 10)$. Note that the IOPT=0 form of F has been normalized by the unknown B(1), so that B(2*MM) is a scaling constant free from B(1); the exact form of B(2*MM) can be determined, if desired, and is related to the constant C in eq. (1) above.

In terms of the NLS notation (Anderson, 1982, p.11-12), let $X(I,1)=t_i$ and Y(I) be the observed F in eq. (4), then the observed data matrix is

(Y(I),X(I,1),I=1,2,...,N).

Since V(t) can range several decades in magnitude for $t_1 \leq t \leq t_N$, it is advised when IOPT=0 that a weighted least-squares option be used (see IWT=1 or 2, Anderson, 1982, p.14-15), which requires the augumented data matrix

(Y(I),X(I,1),X(I,2),I=1,2,...,N),

where X(I,2)=s_i is the standard deviation (IWT=1) of observation Y(I), or X(I,2) is the variance (IWT=2). Note that if s_i is unknown, one could use the statistical weight (Bevington, 1969, p.108) of 1/Y(I) by setting X(I,2)=Y(I) and IWT=2; in this case, this would be preferred over using unity weights (IWT=0). However when IOPT=1, IWT=0 can generally be used since the range of $\rho_a(t)$ is usually much less than the range of V(t) in most cases.

The analytical partial derivative subprogram (PCODE) was not included in program NLSTCI, therefore the estimated derivative option (IDER=1) must be used, which requires only the forward problem solution subprogram (FCODE). See Appendix 3 listing of FCODE for the coding details, which follows the method described in Anderson (1981) for computing V(t), and as revised for computing $\rho_a(t)$.

Because realizable layered earth models are sought to fit the given data, a constrained minimization type (SP=3 or 4) is advised, along with reasonable lower and higher parameter bound arrays, BL(J) and BH(J) respectively, where

$BL(J) \leq B(J) \leq BH(J)$, J=1,2,...,2*MM (see Anderson, 1982, p.17). This approach limits parameter space searching, and in some cases may avoid false starts (or catastrophic overflow conditions from poor estimates and data). In addition, individual parameters can be fixed in the least-squares using parameters IP and IB (Anderson, 1982, p.13). In particular for the IOPT=1 case, one can usually fix B(2*MM)=1, provided the observed (converted) apparent resistivities are properly scaled. Similarly, for the IOPT=0 case, B(2*MM) can be fixed if the constant C in eq.(1) is known a priori. [Actually, if the system calibration is known, then the constant C can be determined; therefore B(2*MM) should be fixed to reduce the number of unknowns, and to reduce the possibility of finding an equivalent but highly improbable solution.] In any case, the user should attempt to give a reasonable starting guess vector B(J) corresponding to the given data matrix. It is advisable to begin with a few layers (e.g., MM=1 or 2) before trying models with more layers. For present TDEM equipment, generally only a few layers are all that can be resolved, due mainly to the small discrete time range $t_1 \leq t \leq t_N$ and noise level in observing V(t).

In general, one should not expect both IOPT=0 and 1 to yield the same exact solutions for a given data set--due mainly to data noise, discrete time-range given, scaling, and the use of different weighting options. For exact data (as in Appendix 2), both IOPT=0 and 1 produce nearly identical solution vectors; for noisy observed data, this is rarely true, although the earth models resolved by both cases should give approximately "equivalent layers" for good fitting cases (i.e., if small parameter errors and RMS error).

PARAMETERS, FILES AND DATA REQUIRED

All <u>$PARMS</u> <u>parameters</u> (excluding the ISTOP=0 option), program files (FOR005-FOR016), and data ordering requirements used by NLSTCI are <u>identical</u> to those described in detail for subprogram NLSOL (Anderson, 1982, p.9-21), and therefore will <u>not</u> be repeated here. However, note that the ordering of the $PARMS estimated parameter vector B(J) used by NLSTCI must be given exactly as described above in eq. (4). The $INIT model parameters required by NLSTCI must be given after the object-time format statement on FOR005 (see Anderson, 1982, p.10, item 5). Also see the EXAMPLE below and Appendix 2 for a typical data input.

$INIT PARAMETER DEFINITIONS

<u>$INIT</u> <u>parameters</u> (nondefault parameters must be given):

MM=	Number of layers in the model ($1 \leq MM \leq 10$; default MM=1 for a homogeneous half-space). Since NLSOL also requires the total number of parameters K, then make sure that K=2*MM is given in $PARMS also. (See the section ERROR MESSAGES below for a discussion on K=2*MM dual input requirement.)
IOPT=0	(default) means that the data matrix

(Y(I),X(I,1),I=1,N) is given with Y(I)=V(t) transient data, which may be unscaled and in any units as determined by B(2*MM) in the least-squares solution. X(I,1)=t_i must be given in <u>seconds</u> and in ascending order for I=1,2,...,N.

IOPT=1 means the data matrix (Y(I),X(I,1),I=1,N) is given with Y(I)=ρ_a(t) apparent resistivity data (in ohm-m.). The shift parameter B(2*MM)=1 can be fixed via $PARMS IP,IB provided the apparent resistivity is known to be scaled correctly. X(I,1)=t_i must be given in <u>seconds</u> and in ascending order for I=1,2,...,N. When IOPT=1 is selected, then ISTEP=0 (see below) can only be used.

A= Radius (in m.) of transmitter circular loop, where A>0 must be given. [Note that a square loop of side L (m.) is considered equivalent to a circular loop of radius A (m.), where A=L/$\sqrt{\pi}$.]

Z= Tranmitter loop elevation (in m.) on or above the surface (default Z=0.0). Note that Z>0 specifies the source loop is Z meters above the surface, but that the central induction receiver coil is assumed to be placed on the surface. For most field applications, Z=0 is always used. [Z is included here only to maintain compatibility with the forward program solution (Anderson, 1981).]

ISTEP=0 (default) to compute the transient derivative response (TDR) sounding (Anderson,1981), which corresponds to the time-derivative of Hz when the source uses a system step driving current (e.g., when using a vertical-axis coil at the loop projected center).

ISTEP=1 to compute the transient field response (TFR) sounding (Anderson, 1981), which corresponds to the integral over time of the TDR-sounding. The TFR-sounding (ISTEP=1) is generally used when transient (stacked) data is obtained using a SQUID or cryogenic magnetometer. Note that Z=0 must be used whenever ISTEP=1.

EPS= Requested convolution integration tolerance used to compute all Fourier and Hankel transforms by digital filtering (default EPS=0.1E-9).

BO=.01 (default) is the lower induction number for which the normalized Hz/DC frequency response approaches the limit 1.0 for B<BO. This assumption saves time by avoiding explicit response calculations for B<BO. BO must be given (or assumed .01 by default) as a power of 10**-n (n integer). The default value is usually adequate for most models; for more accuracy in the late-time transient, BO<.01 can be used. [For accuracy reasons, BO>.01 should never be used.]

BM=100 (default) is the upper induction number for which the normalized Hz/DC frequency response approaches the limit 0.0 for B>BM. This assumption saves time by avoiding explicit response calculations for B>BM. BM must be given (or assumed 100 by default) as a power of 10**n (n integer). The default value is usually quite adequate for most models; for more accuracy in the early-time transient, BM>100 can be used.

NB=8 (default) represents the number of induction number points per decade (log-cycle) to evaluate the pre-splined frequency response function Hz(B)/DC. In general, $3 \leq NB \leq 11$ is usually adequate for most applications (NB<3 is not recommended for accuracy reasons). If NB=0 (or NB>11) is specified, then a direct mode of evaluating the frequency function is used but as controlled by the outer time-integral via lagged convolution (i.e., the cosine filter using subroutine RLAGF0). Note that NB=0 (or NB>11) is more accurate, but much more time-consuming than using NB<12. [See the section COMPUTER TIMING CONSIDERATIONS for a further discussion on the use on NB.]

$END [end of $INIT parameters; the "END" in $END may be omitted, if desired.]

EXAMPLE OF INPUT PARAMETERS AND DATA ORDERING

```
EXAMPLE TITLE WITH OBJECT DATA ON FOR005 (IALT=5)
$PARMS N=20,M=1,K=4,IP=1,IB=4,
IDER=1,IPRT=-1, IALT=5,SP=3,IWT=1, NITER=5,
BL=2*.0001,10,.1, B=.1,.01,100,.1,
BH=2*10,1000,.1$
(3F10.0)
0.1        .0004     .18
0.03       .0008     .09
---<etc. for 18 more observations>---
$INIT MM=2,A=100,NB=4,EPS=.1E-5$END
```

(See Appendix 2 for a complete input/output example.)

COMPUTER TIMING CONSIDERATIONS

The computer CPU-time will vary mostly as a function of the given $INIT parameters MM,EPS,BO,BM,NB and $PARMS parameters N,NITER,IP,SP,IV,V, and B. Perhaps the parameters of greatest effect on CPU-time are how good the initial model estimates are given in array B(J), J=1,2,...,2*MM, with respect to the observed data matrix. Of course, the observed data matrix time-range and noise level can contribute further problems in resolving a given layered earth model for any MM in (1,10). In some cases, it may be necessary to fix certain parameters in B (via $PARMS IP,IB) that cannot be resolved and/or to control the initial theoretical transient curve behavior. Generally, it is best to begin with a small MM (say 2 or 3), and progressively increase MM until the RMS error cannot be further decreased. During this "initial model searching study", several $INIT parameters can be modified (relaxed) to significantly reduce the overall CPU-time, but with somewhat less accurate results (which may not be needed for initial runs). Some suggestions are provided in Table 1.

Table 1. Recommended $INIT parameters for NLSTCI

$INIT parameter	Default value	Faster CPU; less accurate	Slower CPU; more accurate
EPS	0.1E-9	0.1E-5	0.1E-11
BO	0.01	0.01*	0.001
BM	100	10	1000
NB	8	2<NB<8	8<NB<12

* BO>0.01 should never be used
(see Anderson, 1981, p. 25-41).

For a final model run, the default values in Table 1 are generally sufficient for most field situations, with the exception that NB>8 may be used to reduce any noticeable nonsmoothness in the calculated transient. (Note that NB>11 is not recommended for routine field work.)

Some $PARMS parameters used in the NLS algorithm can also be modified to reduce the total CPU-time when searching for an initial model. In particular, $PARMS NITER (Anderson, 1982, p. 16) can be set small (e.g., 3 or 5) to force termination of a trial run after just a few iterations. This is reasonable, since it may not be necessary to obtain normal convergence of the iteration process for preliminary or intermediate models. Other $PARMS that control the NLS algorithm speed and accuracy can also be overridden from their default values (see Table 2 in Anderson, 1982, p. 20-21 for more details).

DATA MATRIX NOTES

The data matrix (defined following eq. (4)) is read under the object-time format statement, and is defined as the sequence of ordered rows:

(Y(I),(X(I,L),L=1,M*),I=1,N),

where M*=M if IWT=0 (default), or M*=M+1 if IWT=1 or 2. In the above example, IWT=1, M=1, and therefore three columns are required in the data matrix row, where in this case, the last column represents the standard deviation of observation Y(I).

SPECIAL OBJECT FORMAT PHRASES

If an existing data matrix file does not have the proper defined column ordering in the form (Y(I),X(I,J),J=1,M), then the FORTRAN "Tn" format phrase may be used to begin at any column n in the data record. For example, the format (T41,F10.0,T1,2F10.0) will select Y(I) using column 41-50 and X(I,1) beginning at column 1. See any FORTRAN-77 coding manual for other allowable object (run) time format phrases (e.g., the G-format, use of "/" to skip records, etc.). Note that "tab"-characters must not be used when creating the data matrix file FOR010.

VAX OPERATING INSTRUCTIONS

In general, the basic steps described to run NLSOL (Anderson, 1982, p.22-24) can be followed to run NLSTCI either on-line or in batch mode. That is, the parameter and data matrix files may be associated with the logical names FOR005 and FOR010, respectively, using the VAX-DCL statements:

```
     $ASSIGN parameterfilename FOR005
     $ASSIGN datamatrixfilename FOR010
     $RUN NLSTCI !(use $RUN [WANDERSON]NLSTCI on USGS VAX)
```

If the data matrix is included on FOR005 (i.e., using IALT=5), then the FOR010 assignment is not necessary.

In addition, program NLSTCI has a useful "restart file" (called FOR005.TMP) that is automatically provided each time the program is executed. File FOR005.TMP contains a copy of all parameters on FOR005, plus the last solution B-vector obtained; note that $PARMS ISTOP=0 (Anderson, 1982, p.14) cannot be used because FOR005 is positioned at EOF in creating FOR005.TMP. If desired, one can easily continue (or restart) more iterations simply by using the DCL commands:

```
$ASSIGN FOR005.TMP FOR005
$RUN NLSTCI !(use $RUN [WANDERSON]NLSTCI on USGS VAX)
```

Note that FOR005.TMP may also be edited (using any VAX editor) for other parameter changes, if desired. Also, the reassignment of FOR005 using FOR005.TMP only needs to be done once for multiple continuation runs.

By default, the master print (disk) file is called FOR016.DAT, unless otherwise assigned. This file can be TYPEd or PRINTed on a line printer. Also, file FOR016 may be used as an input file to a plot routine; e.g., to plot the observed (OBS), calculated (CAL), and residual (RES) curves. If program NLSTCI is run on-line, then a shorter terminal print file on FOR006 contains some of the information as on FOR016, but as controlled by parameter IPRT (Anderson, 1982, p.15).

ERROR MESSAGES

Almost all $PARMS syntactical errors are flagged and printed on files FOR006 and FOR016 and the job is aborted (see Anderson, 1982, p.24). However, some cross-references (or dual inputs) are not checked; for example, the relationship K=2*MM is not double checked between $PARMS K and $INIT MM parameters. This is because a general-purpose nonlinear least-squares algorithm (NLSOL) is being used as a control program, but the model input is external to the particular nonlinear problem requirements (NLSTCI) read by subprogram SUBZ (see Anderson, 1982, p.38). Therefore, the user is responsible for providing exactly K parameter estimates in B(I),I=1,2,...,K (see eq. (4)), and that $INIT MM is such that K=2*MM (otherwise, unpredictable results could occur).

The message "{WARN}: NOISE IN CALC. TRANS DETECTED" can occur for certain model estimates in array B with respect to the given data matrix. This warning message actually means that the calculated transient voltage V/I cannot be computed accurately at late times using single-precision arithmetic (regardless of the values specified in $INIT parameters EPS,BO,BM, and NB). However, this condition is usually unimportant if the warning occurs near the beginning of the NLS iteration. For typical field data cases, and a moderate MM value and reasonable B estimates, one should not expect the warning message to appear near the end of the NLS iterations for a converging model solution.

PRINTED OUTPUT

All input parameters are output on files FOR006 and FOR016, with the $INIT parameters given first, followed by all $PARMS parameters given or assumed by default. (Refer to Appendix 2 for a complete sample output listing.)

Specific names (e.g., IT, NF, ...) used by NLSOL in the output listings are tabulated in Anderson (1982, p.25-26). Program NLSTCI provides a summary listing of the final solution vector B, along with accumulated layer thicknesses listed under the DEPTH column (see the end of the listing example in Appendix 2). The RESISTIVITY column is simply 1/SIGMA, where SIGMA is the layer conductivity (in mhos/m.).

REFERENCES

Anderson, W.L., 1975, Improved digital filters for evaluating Fourier and Hankel transform integrals: USGS Rept. GD-75-012, 223 p. (also available as NTIS Rept. PB-242-800.)

--------, 1979, Numerical integration of related Hankel transforms of orders 0 and 1 by adaptive digital filtering: Geophysics, v. 44, n. 7, p. 1287-1305.

--------, 1981, Calculation of transient soundings for a central induction loop system (Program TCILOOP): USGS Open-File Rept. 81-1309, 80 p.

--------, 1982, Adaptive nonlinear least-squares solution for constrained or unconstrained minimization problems

(Subprogram NLSOL): USGS Open-File Rept. 82-68, 65 p.

Bevington, P.R., 1969, Data reduction and error analysis for the physical sciences: McGraw-Hill, N.Y., 336 p.

Dennis, J.E., Gay, D.M., and Welsch R.E., 1979, An adaptive nonlinear least-squares algorithm: Univ. of Wisconsin MRC Tech. Sum. Rept. 2010 (also available as NTIS Rept. AD-A079-716),

Raab, P.V., and Frischknecht, F.C., 1982, Desk top programs for central and coincident loop TDEM calculations: USGS Open-File Rept. 82-xxx, yy p [in press].

Appendix 1.-- Conversion to other systems

This program (and associated subprograms) was written in ANSI-standard FORTRAN-77 for the VAX-11/780 system (VMS version 2.5). Conversion to systems without an ANSI-FORTRAN-77 compiler would necessitate extensive changes, particularly for all CHARACTER-type variables, IF-THEN-ELSE phrases, etc.

Since the FORTRAN-77 ANSI-standard presently does not provide for a NAMELIST I/O capability, a VAX-11 NAMELIST simulator subprogram is included in this program package. For most large main-frame systems (e.g., IBM/370, CYBER, etc.), a NAMELIST READ/WRITE is usually available; in this case, the VAX NAMELIST subprogram and associated routines (DECODEIX, DECODEX) can be eliminated; also, appropriate changes can be made where COMMON/NAME_LIST/ and CALL NAMELIST is used in the source program.

Other changes for non-VAX systems might include some (or all) of the following:

(1) Variables with more than 6-characters.

(2) Use of the underscore character or dollar character in some variables and/or COMMON names.

(3) Character strings delimited by single-quote characters (e.g., 'STRING'); also, character string concatination (e.g., 'STRING1'//'STRING2').

(4) Passing variable-length character strings in subroutine calls; e.g., CHARACTER*(*) passed length character

arguments.

(5) Need to suppress arithmetic or exponential underflow messages (note that a VAX-11 result is automatically set to 0.0 after any underflow--which is assumed for this program package); if the target system does not set underflows to 0.0 (and suppress warning messages), then a suitable conversion procedure must be used for proper operation of this program package.

(6) Replacement of any special VAX-dependent CALLS or statements (e.g., CALL LIB$INDEX, ACCEPT, TYPE, CALL SYS$anyname, etc.--note that we have minimized machine-dependent calls, where possible).

(7) Hexidecimal constants (e.g., '4A'X) if used in any DATA statements.

(8) Virtual-sized arrays, if any (i.e, DIMENSION statements greater than physical memory).

Appendix 2.-- Test problem input/output listing

The following input files (FOR005.0, FOR010, FOR005.1) were used to run a known test problem for program NLSTCI on a VAX system using both IOPT=0 and 1 cases separately. The corresponding output files (FOR016) are given following FOR005.1. In addition, each file FOR016.DAT was used to plot the final observed (OBS) and calculated (CAL) curves using an external plotter. The symbol "O" represents Y(I) in the plot, and the solid line represents a curve drawn through the calculated (CAL) points.

FOR005.0

```
TEST EXAMPLE (IOPT=0 CASE)
$PARMS N=19,K=4,M=1,IPRT=-2,
IDER=1,IWT=2,SP=3,
NITER=15,
BL=2*.0001,10,.1E-4,
B=.015,.15,175,.015,
BH=2*5,1000,.1E5$
(2G16.8,T1,G16.8)
$INIT MM=2,A=175$
```

FOR010

```
  0.96605726E-01  0.19242254E-03  0.11366887E+03
  0.36346640E-01  0.28243766E-03  0.12427132E+03
  0.13687826E-01  0.41456183E-03  0.13160466E+03
  0.64035309E-02  0.60849357E-03  0.11748647E+03
  0.35103841E-02  0.89314644E-03  0.93367012E+02
  0.21471761E-02  0.13109597E-02  0.68623184E+02
  0.13093358E-02  0.19242257E-02  0.50529686E+02
  0.78840711E-03  0.28243773E-02  0.37524563E+02
  0.45996808E-03  0.41456190E-02  0.28474813E+02
  0.25708214E-03  0.60849362E-02  0.22246769E+02
  0.13831072E-03  0.89314654E-02  0.17830414E+02
  0.71406452E-04  0.13109598E-01  0.14687531E+02
  0.35465542E-04  0.19242259E-01  0.12410449E+02
  0.16997505E-04  0.28243775E-01  0.10733011E+02
  0.78670519E-05  0.41456193E-01  0.94950829E+01
  0.35501544E-05  0.60849369E-01  0.85372982E+01
  0.15584020E-05  0.89314662E-01  0.78144689E+01
  0.66868074E-06  0.13109601E+00  0.72586303E+01
  0.28317137E-06  0.19242261E+00  0.67987070E+01
```

FOR005.1

```
TEST EXAMPLE (IOPT=1 CASE)
$PARMS N=19,K=4,M=1,IPRT=-2,
IDER=1,IWT=0,SP=3,
NITER=15,IP=1,IB=4,
BL=2*.0001,10,.1E-4,
B=.015,.15,175,1,
BH=2*5,1000,.1E5$
(T33,G16.8,T17,G16.8)
$INIT MM=2,A=175,IOPT=1$
```

FOR016

```
{NLSTCI}:     TEST EXAMPLE (IOPT=0 CASE)

MM=  2              A= 0.175000E+03    EPS= 0.100000E-09
BO= 0.100000E-01  BM= 0.100000E+03    NB=  8
Z= 0.000000E+00   ISTEP=  0
IOPT=  0



PARAMETER ORDER--

      1       SIGMA(  1)
      2       SIGMA(  2)
      3       THICK(  1)
      4           B(  4) SHIFT PARAMETER IN B(2*MM)*TRANSIENT
```

```
{NLSOL}:        TEST EXAMPLE (IOPT=0 CASE)

N=          19    K=          4    IP=         0    M=          1    IALT=      10
ISTOP=       1    IWT=        2    IDER=       1    IPRT=      -2    NITER=     15
IOUT=        1    SP=         3

FMT=(2G16.8,T1,G16.8)


PARAMETER LOWER BOUNDS: BL=

 0.99999997E-04  0.99999997E-04  0.10000000E+02  0.99999997E-05

INITIAL PARAMETERS: B=

 0.15000000E-01  0.15000001E+00  0.17500000E+03  0.15000000E-01

PARAMETER HIGHER BOUNDS: BH=

 0.50000000E+01  0.50000000E+01  0.10000000E+04  0.10000000E+05

** NLITR (IDER=0) OR NL2SNO (IDER=1) CALLED:  1 **


     I      INITIAL X(I)       D(I)

     1      0.546171E-01     0.490E+02
     2      0.174026E+00     0.106E+01
     3      0.420534E+00     0.230E+01
     4      0.122434E-02     0.186E+04

    IT     NF       F           DF          COSMAX       VAR

     0      1  0.200E+00               0.999E+00
     1      2  0.110E-01   0.189E+00   0.994E+00   0.150E+02
     2      3  0.831E-04   0.109E-01   0.924E+00   0.150E+02
     3      4  0.756E-05   0.755E-04   0.946E+00   0.149E+02
     4      5  0.324E-06   0.724E-05   0.655E+00   0.145E+02
     5      6  0.118E-06   0.206E-06   0.656E+00   0.140E+02
     6      7  0.225E-07   0.959E-07   0.686E+00   0.123E+02
     7      8  0.909E-08   0.135E-07   0.911E+00   0.141E+02
     8      9  0.650E-10   0.903E-08   0.995E-01   0.151E+02
     9     10  0.636E-10   0.138E-11   0.186E+00   0.144E+01
    10     11  0.433E-10   0.202E-10   0.439E+00   0.146E+01
    11     12  0.433E-10  -0.117E-10   0.439E+00   0.818E+01

***** X-CONVERGENCE *****

FUNCTION      0.433499D-10   VARIABILITY    0.818060E+01
FUNC. EVALS       12         GRAD. EVALS        11
GRAD. NORM    0.104053E-02   COSMAX         0.438530E+00

     I        FINAL X(I)        D(I)           G(I)

     1      0.445184E-01     0.430E+02      0.404E-04
     2      0.201355E+00     0.426E+00      0.151E-05
     3      0.453441E+00     0.162E+01      0.504E-05
     4      0.999343E-03     0.131E+04      0.104E-02

COVARIANCE = SCALE * (J**T * J)**-1

ROW  1     0.3025E-11
```

FOR005.1

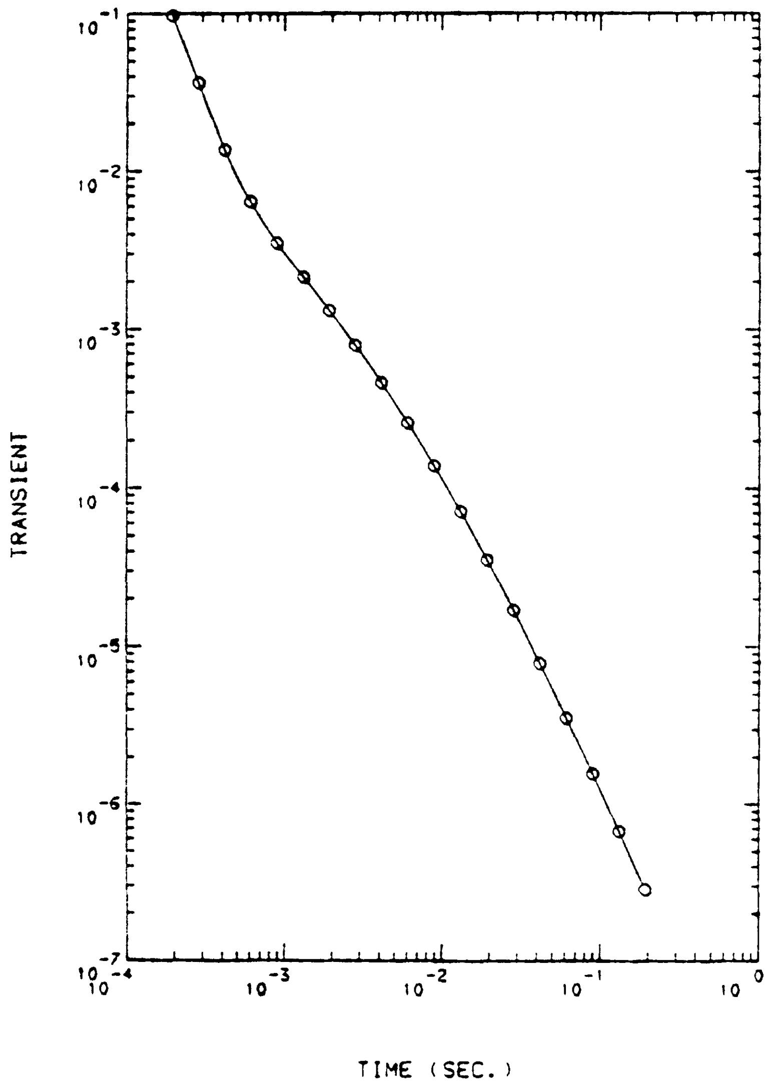
```
TEST EXAMPLE (IOPT=1 CASE)
$PARMS N=19,K=4,M=1,IPRT=-2,
IDER=1,IWT=0,SP=3,
NITER=15,IP=1,IB=4,
BL=2*.0001,10,.1E-4,
B=.015,.15,175,1,
BH=2*5,1000,.1E5$
(T33,G16.8,T17,G16.8)
$INIT MM=2,A=175,IOPT=1$
```

FOR016

```
{NLSTCI}:      TEST EXAMPLE (IOPT=0 CASE)

MM=  2               A= 0.175000E+03    EPS= 0.100000E-09
B0= 0.100000E-01  BM= 0.100000E+03    NB=  8
Z= 0.000000E+00   ISTEP=  0
IOPT=  0



PARAMETER ORDER--

      1       SIGMA(   1)
      2       SIGMA(   2)
      3       THICK(   1)
      4           B(   4) SHIFT PARAMETER IN B(2*MM)*TRANSIENT
```

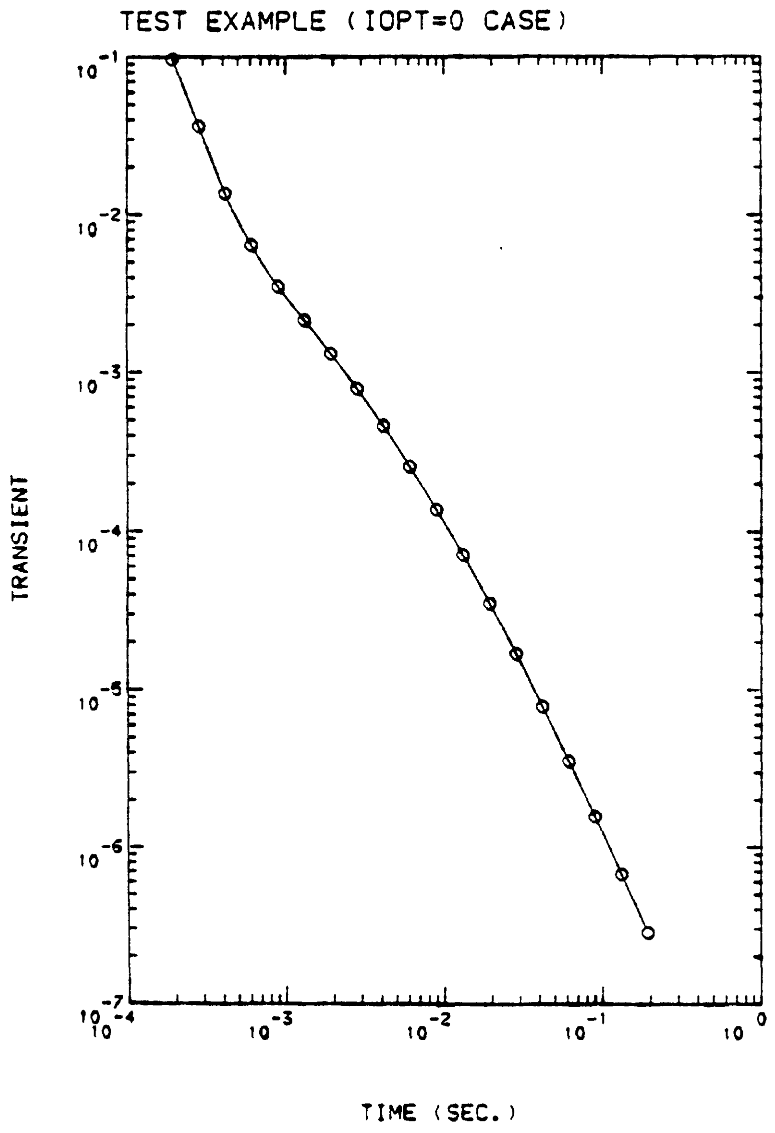
```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TOTAL "ELAPSED" TIME=           282.81 SEC. (    4 MIN. 42.81 SEC.)
CPU_TIME=          263.12 SEC. (    4 M. 23.12 S.)    CPU % = 93.04%
BUF.I/O_COUNT=          7
DIR.I/O_COUNT=         20
PAGE_FAULTS=          148
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

TEST EXAMPLE (IOPT=0 CASE)



```
ROW  2     0.2246E-10  0.2355E-09
ROW  3    -0.1491E-10 -0.1127E-09  0.7896E-10
ROW  4    -0.7822E-13 -0.5524E-12  0.3802E-12  0.2047E-14

  I    OBS.Y(I)      CAL             RES       %RES.ERR      X(I,1)        X(I,2)        X(I,3)        X(I,4)         WT(I)
  1  0.966057E-01  0.966070E-01 -0.132E-05 -0.136507E-02  0.192423E-03  0.966057E-01  0.000000E+00  0.000000E+00  0.103514E+02
  2  0.363466E-01  0.363465E-01  0.156E-06  0.430474E-03  0.282438E-03  0.363466E-01  0.000000E+00  0.000000E+00  0.275129E+02
  3  0.136878E-01  0.136875E-01  0.278E-06  0.203444E-02  0.414562E-03  0.136878E-01  0.000000E+00  0.000000E+00  0.730576E+02
  4  0.640353E-02  0.640327E-02  0.262E-06  0.408700E-02  0.608494E-03  0.640353E-02  0.000000E+00  0.000000E+00  0.156164E+03
  5  0.351038E-02  0.351037E-02  0.158E-07  0.451021E-03  0.893146E-03  0.351038E-02  0.000000E+00  0.000000E+00  0.284869E+03
  6  0.214718E-02  0.214698E-02  0.196E-06  0.910943E-02  0.131096E-02  0.214718E-02  0.000000E+00  0.000000E+00  0.465728E+03
  7  0.130934E-02  0.130925E-02  0.851E-07  0.649987E-02  0.192423E-02  0.130934E-02  0.000000E+00  0.000000E+00  0.763746E+03
  8  0.788407E-03  0.788485E-03 -0.777E-07 -0.985526E-02  0.282438E-02  0.788407E-03  0.000000E+00  0.000000E+00  0.126838E+04
  9  0.459968E-03  0.459998E-03 -0.295E-07 -0.642186E-02  0.414562E-02  0.459968E-03  0.000000E+00  0.000000E+00  0.217406E+04
 10  0.257082E-03  0.257110E-03 -0.277E-07 -0.107876E-01  0.608494E-02  0.257082E-03  0.000000E+00  0.000000E+00  0.388981E+04
 11  0.138311E-03  0.138347E-03 -0.358E-07 -0.259069E-01  0.893147E-02  0.138311E-03  0.000000E+00  0.000000E+00  0.723010E+04
 12  0.714065E-04  0.714202E-04 -0.137E-07 -0.191831E-01  0.131096E-01  0.714065E-04  0.000000E+00  0.000000E+00  0.140043E+05
 13  0.354655E-04  0.354660E-04 -0.469E-09 -0.132324E-02  0.192423E-01  0.354655E-04  0.000000E+00  0.000000E+00  0.281964E+05
 14  0.169975E-04  0.170017E-04 -0.416E-08 -0.244898E-01  0.282438E-01  0.169975E-04  0.000000E+00  0.000000E+00  0.588322E+05
 15  0.786705E-05  0.786947E-05 -0.241E-08 -0.306730E-01  0.414562E-01  0.786705E-05  0.000000E+00  0.000000E+00  0.127112E+06
 16  0.355015E-05  0.355112E-05 -0.963E-09 -0.271226E-01  0.608494E-01  0.355015E-05  0.000000E+00  0.000000E+00  0.281678E+06
 17  0.155840E-05  0.155997E-05 -0.157E-08 -0.100717E+00  0.893147E-01  0.155840E-05  0.000000E+00  0.000000E+00  0.641683E+06
 18  0.668681E-06  0.668978E-06 -0.298E-09 -0.445075E-01  0.131096E+00  0.668681E-06  0.000000E+00  0.000000E+00  0.149548E+07
 19  0.283171E-06  0.283069E-06  0.103E-09  0.362866E-01  0.192423E+00  0.283171E-06  0.000000E+00  0.000000E+00  0.353143E+07

** RMSERR=  0.36189712E-06

CORRELATION MATRIX
 1  0.1000E+01
 2  0.8414E+00  0.1000E+01
 3 -0.9648E+00 -0.8264E+00  0.1000E+01
 4 -0.9940E+00 -0.7957E+00  0.9458E+00  0.1000E+01

  **PARM_SOL.   STD_ERROR   REL_ERROR    % ERROR **

 1  0.1000E-01  0.1739E-05  0.1739E-03  0.1739E-01
 2  0.2001E+00  0.1535E-04  0.7670E-04  0.7670E-02
 3  0.2000E+03  0.8886E-05  0.4443E-07  0.4443E-05
 4  0.9997E-02  0.4524E-07  0.4525E-05  0.4525E-03


********  E N D  ********     TEST EXAMPLE (IOPT=0 CASE)

PARAMETER NAME      FINAL SOLUTION       RESISTIVITY   LAYER DEPTH

   1   SIGMA( 1) =  0.10002709E-01   1  0.99972916E+02
   2   SIGMA( 2) =  0.20009081E+00   2  0.49977307E+01
   3   THICK( 1) =  0.19997893E+03                       1  0.19997893E+03
   4   SHIFT     =  0.99968594E-02
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TOTAL "ELAPSED" TIME=          282.81 SEC. (    4 MIN. 42.81 SEC.)
CPU_TIME=          263.12 SEC. (    4 M. 23.12 S.)    CPU % = 93.04%
BUF.I/O_COUNT=           7
DIR.I/O_COUNT=          20
PAGE_FAULTS=           148
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

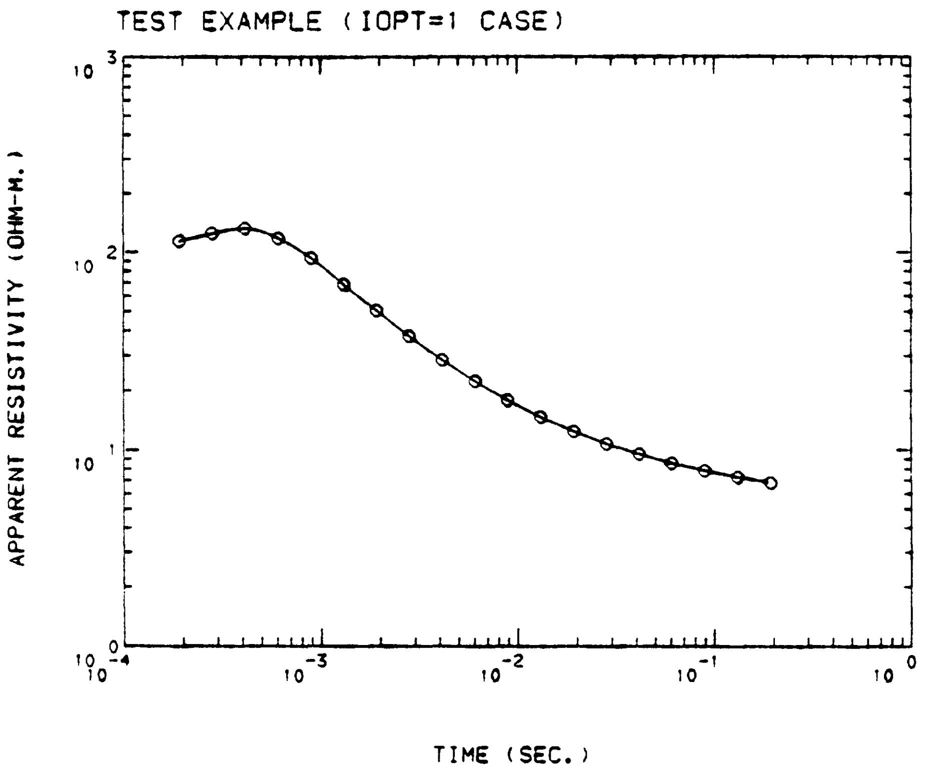


```
{NLSTCI}:     TEST EXAMPLE (IOPT=1 CASE)

MM=  2              A= 0.175000E+03    EPS= 0.100000E-09
B0= 0.100000E-01  BM= 0.100000E+03    NB=  8
Z= 0.000000E+00   ISTEP=  0
IOPT=  1



PARAMETER ORDER--

      1       SIGMA(  1)
      2       SIGMA(  2)
      3       THICK(  1)
      4           B(  4) SHIFT PARAMETER IN B(2*MM)*APPRES
```

```
{NLSOL}:        TEST EXAMPLE (IOPT=1 CASE)

N=          19   K=           4   IP=          1   M=            1   IALT=       10
ISTOP=       1   IWT=         0   IDER=        1   IPRT=        -2   NITER=      15
IOUT=        1   SP=          3

PARAMETERS HELD FIXED: IB=  4

FMT=(T33,G16.8,T17,G16.8)


PARAMETER LOWER BOUNDS: BL=

 0.99999997E-04  0.99999997E-04  0.10000000E+02  0.99999997E-05

INITIAL PARAMETERS: B=

 0.15000000E-01  0.15000001E+00  0.17500000E+03  0.10000000E+01

PARAMETER HIGHER BOUNDS: BH=

 0.50000000E+01  0.50000000E+01  0.10000000E+04  0.10000000E+05

PARAMETER INDEX:  1  2  3  4
REORDERED AS...:  1  2  3

REORDERED PARAMETERS:

 0.15000000E-01  0.15000001E+00  0.17500000E+03

** NLITR (IDER=0) OR NL2SNO (IDER=1) CALLED:  1 **


    I      INITIAL X(I)       D(I)

    1      0.546171E-01     0.638E+04
    2      0.174026E+00     0.355E+03
    3      0.420534E+00     0.518E+03

   IT     NF        F            DF          COSMAX       VAR

    0      1  0.468E+04                   0.984E+00
    1      2  0.186E+04  0.282E+04  0.993E+00  0.159E+02
    2      3  0.478E+02  0.181E+04  0.814E+00  0.159E+02
    3      4  0.420E+00  0.474E+02  0.720E+00  0.157E+02
    4      5  0.234E-02  0.418E+00  0.782E+00  0.150E+02
    5      6  0.383E-04  0.231E-02  0.240E+00  0.155E+02
    6      7  0.383E-04 -0.602E-04  0.240E+00  0.118E+01

***** X-CONVERGENCE *****

FUNCTION      0.383166D-04  VARIABILITY    0.118399E+01
FUNC. EVALS        7        GRAD. EVALS         6
GRAD. NORM    0.720512E+01  COSMAX         0.239794E+00

    I        FINAL X(I)        D(I)           G(I)

    1      0.445123E-01     0.125E+05     -0.697E+01
    2      0.201312E+00     0.393E+03      0.254E-01
    3      0.453472E+00     0.874E+03      0.183E+01

COVARIANCE = SCALE * (J**T * J)**-1
```

```
ROW  1     0.5643E-13
ROW  2     0.1198E-11  0.5670E-10
ROW  3    -0.1210E-12 -0.2919E-11  0.6546E-11

 I    OBS.Y(I)        CAL           RES        %RES.ERR       X(I,1)        X(I,2)        X(I,3)        X(I,4)        WT(I)
 1  0.113669E+03  0.113669E+03  0.107E-03  0.939673E-04  0.192423E-03  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
 2  0.124271E+03  0.124270E+03  0.900E-03  0.724443E-03  0.282438E-03  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
 3  0.131605E+03  0.131603E+03  0.204E-02  0.155368E-02  0.414562E-03  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
 4  0.117486E+03  0.117492E+03 -0.530E-02 -0.451302E-02  0.608494E-03  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
 5  0.933670E+02  0.933657E+02  0.130E-02  0.138916E-02  0.893146E-03  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
 6  0.686232E+02  0.686239E+02 -0.717E-03 -0.104506E-02  0.131096E-02  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
 7  0.505297E+02  0.505287E+02  0.973E-03  0.192514E-02  0.192423E-02  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
 8  0.375246E+02  0.375267E+02 -0.216E-02 -0.574339E-02  0.282438E-02  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
 9  0.284748E+02  0.284767E+02 -0.188E-02 -0.659746E-02  0.414562E-02  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
10  0.222468E+02  0.222456E+02  0.117E-02  0.524732E-02  0.608494E-02  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
11  0.178304E+02  0.178293E+02  0.113E-02  0.632242E-02  0.893147E-02  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
12  0.146875E+02  0.146877E+02 -0.154E-03 -0.104538E-02  0.131096E-01  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
13  0.124104E+02  0.124118E+02 -0.135E-02 -0.108877E-01  0.192423E-01  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
14  0.107330E+02  0.107327E+02  0.283E-03  0.263904E-02  0.282438E-01  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
15  0.949508E+01  0.949251E+01  0.258E-02  0.271459E-01  0.414562E-01  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
16  0.853730E+01  0.853363E+01  0.367E-02  0.429921E-01  0.608494E-01  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
17  0.781447E+01  0.781489E+01 -0.420E-03 -0.537555E-02  0.893147E-01  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
18  0.725863E+01  0.726051E+01 -0.188E-02 -0.258827E-01  0.131096E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01
19  0.679871E+01  0.680066E+01 -0.196E-02 -0.287617E-01  0.192423E+00  0.000000E+00  0.000000E+00  0.000000E+00  0.100000E+01

** RMSERR=  0.21885093E-02

ORRELATION MATRIX
 1  0.1000E+01
 2  0.6699E+00  0.1000E+01
 3 -0.1991E+00 -0.1515E+00  0.1000E+01

  **PARM_SOL.   STD_ERROR   REL_ERROR    % ERROR **

 1  0.1000E-01  0.2376E-06  0.2376E-04  0.2376E-02
 2  0.2000E+00  0.7530E-05  0.3765E-04  0.3765E-02
 3  0.2000E+03  0.2559E-05  0.1279E-07  0.1279E-05


********   E N D   ********      TEST EXAMPLE (IOPT=1 CASE)

ARAMETER NAME      FINAL SOLUTION       RESISTIVITY   LAYER DEPTH

  1    SIGMA( 1) =  0.10000006E-01    1  0.99999939E+02
  2    SIGMA( 2) =  0.20000516E+00    2  0.49998713E+01
  3    THICK( 1) =  0.20000261E+03                         1  0.20000261E+03
  4    SHIFT     =  0.10000000E+01
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TOTAL "ELAPSED" TIME=           184.76 SEC. (   3 MIN.  4.76 SEC.)
CPU_TIME=            167.67 SEC. (   2 M. 47.67 S.)    CPU % = 90.75%
BUF.I/O_COUNT=           7
DIR.I/O_COUNT=          19
PAGE_FAULTS=           153
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```



Appendix 3.-- Source code availability and listing

Source Code Availability

The current version of the source code may be obtained by writing directly to the author*. A magnetic tape copy can be sent to requestors to be copied and returned. This method of releasing the source code was selected in order to satisfy requests for the latest (e.g., possibly updated) version. [The attached listing does not include the adaptive nonlinear least-squares algorithm (Dennis and others, 1979) due to its length; however, the complete algorithm is available on the distributed tape.]

The magnetic tape is usually recorded in the following mode (unless requested otherwise):

> Industry compatible: 9-track, standard ANSI-labeled, ASCII-mode, odd-parity, 800-bpi density, 80-character card-image records (blocked 50-card images, or 4000-characters, per physical block), and contained on a file named "NLSTCI.VAX".

------
* present address is:

U.S. Geological Survey
Mail Stop 964
Box 25046, Denver Federal Center
Denver, CO 80225

Source Listing

The attached subprograms are listed in the following order:

```
C (NLSTCI): 'NLSOL'-INVERSION OF TRANSIENT SOUNDINGS FOR  (8/9/82)     00000010
C  A CENTRAL INDUCTION LOOP SYSTEM OF RADIUS A>0.                      00000020
C                                                                      00000030
C** VAX-11/780 VERSION                                                 00000040
C                                                                      00000050
C--BY W.L.ANDERSON, U.S. GEOLOGICAL SURVEY, DENVER, COLORADO.          00000060
C                                                                      00000070
C                                                                      00000080
      EXTERNAL MARQ_TRANS_HZLOOP_FCODE,DUMYPCODE,                      00000090
     1 MARQ_TRANS_HZLOOP_SUBZ,SIGSUBEND                                00000100
      CALL SETTIME                                                     00000110
      CALL NLSOL(MARQ_TRANS_HZLOOP_FCODE,DUMYPCODE,                    00000120
     1 MARQ_TRANS_HZLOOP_SUBZ,SIGSUBEND)                               00000130
      CALL CPUTIME(6,16)                                               00000140
      CALL EXIT                                                        00000150
      END                                                              00000160
      REAL FUNCTION HZLOOP(B2)                                         00000170
C--COSINE-TRANSFORM KERNEL FOR CENTRAL INDUCTION LOOP WITH             00000180
C  A>0,R=0, AND Z>=0.0.                                                00000190
C                                                                      00000200
      REAL SIG(10),H(10),Z                                             00000210
      COMPLEX ZHANKS,ZAC4,K2(10),KS1,ZFLD                              00000220
      COMMON/MODEL/K2,KS1,H,Z,A,R,HMAX,M                               00000230
      COMMON/PASS/ZAC4,ANORM,CURI,DC,SIG,B0,BM,SIG1,EPS,ISTEP          00000240
      COMMON/SPLN/XS(200),YS(200),AS(200),BS(200),CS(200),NS,ISPLN     00000250
      EXTERNAL F3ZH                                                    00000260
      B=SQRT(B2)                                                       00000270
      IF(B.LT.B0) GO TO 3                                              00000280
      IF(B.GT.BM) GO TO 4                                              00000290
      IF(ISPLN.EQ.0) GO TO 10                                          00000300
C--ISPLN=1 (0<NB<12 OPTION) INTERPOLATE PRE-SPLINED FREQ. FUNCTION     00000310
      CALL SPOINT(NS,XS,YS,AS,BS,CS,B,HZLOOP)                          00000320
      RETURN                                                           00000330
```

```
10    F=(B/A)**2/(39.47841762E-7*SIG1)
      KS1=CMPLX(0.0,-7.895683523E-6*F)
      DO 1 I=1,M
1     K2(I)=KS1*CMPLX(SIG(I),0.0)
      ZFLD=ANORM*ZHANKS(1,ANORM,F3ZH,EPS,LL,1) + ZAC4
      ZFLD=CMPLX(CURI,0.0)*ZFLD
      HZLOOP=REAL(ZFLD)/DC
      RETURN
3     HZLOOP=1.0
      RETURN
4     HZLOOP=0.0
      RETURN
      END
      COMPLEX FUNCTION F3ZH(X)
C--KERNEL FOR HANKEL TRANSFORM IN CURLOOP WHEN R=0.0 AND Z>=0.0
C  SCALED BY HMAX STORED IN COMMON/MODEL/
C
      COMPLEX Z1,Z0,K2(10),KS1,HALF
      REAL H(10),Z
      COMMON/MODEL/K2,KS1,H,Z,A,R,HMAX,M
      DATA HALF/(0.5,0.0)/
      Y=X/HMAX
      CALL RECUR(Y,Z1,Z0)
      F3ZH=CMPLX(Y,0.0)*(Z1/(Z0+Z1)-HALF)
      IF(Z.GT.0.0) F3ZH=F3ZH*CMPLX(EXP(-Y*Z),0.0)
      RETURN
      END
      SUBROUTINE RECUR(Y,Z1,Z0)
C--BACKWARD RECURRENCE FOR COMPLEX IMPEDANCES Z1,Z0 GIVEN ARGUMENT
C     Y(=X/HMAX) AND MODEL PARAMETERS IN COMMON/MODEL/
C
      REAL H(10),Z
      COMPLEX Z1,Z0,K2(10),KS1,ONE,ZZ,X2,U
      COMMON/MODEL/K2,KS1,H,Z,A,R,HMAX,M
      DATA ONE/(1.0,0.0)/
      X2=CMPLX(Y*Y,0.0)
      Z0=KS1/CMPLX(Y,0.0)
      Z1=KS1/CSQRT(X2-K2(M))
      IF(M.EQ.1) GO TO 20
      J=M-1
10    U=CSQRT(X2-K2(J))
      ZZ=KS1/U
      U=CEXP(CMPLX(-2.0*H(J),0.0)*U)
      U=(ONE-U)/(ONE+U)
      Z1=ZZ*((Z1+ZZ*U)/(ZZ+Z1*U))
      IF(J.EQ.1) GO TO 20
      J=J-1
      GO TO 10
20    RETURN
      END
      SUBROUTINE MARQ_TRANS_HZLOOP_FCODE(Y,X,B,PRNT,F,IN,IDER)
C--FUNCT. EVAL. FOR 'NLSTCI'
C
C--PARAMETERS--
C      Y=      OBSERVED DEPENDENT VARIABLE ARRAY (DIM. N)
```

```
C       X=       OBSERVED INDEPENDENT VARIABLE ARRAY (DIM. N,5)
C       B=       CURRENT PARAMETER ARRAY ESTIMATES (DIM. K)
C       PRNT=    WORK AND PRINT ARRAY (DIM. 5)
C       F=       OUTPUT FUNCTION VALUE EVAL. FOR GIVEN Y,X,B AT OBS. IN
C       IN=      OBSERVATION NO. TO EVAL. F (1<=IN<=N)
C       IDER=    0 IF ANALYTIC DERIVATIVES ARE USED LATER (PCODE CALLED)
C                1 IF ESTIMATED DERIVATIVES USED ONLY (PCODE NOT CALLED)
C [NOTE: CURRENTLY ONLY IDER=1 CAN BE USED; IDER=0 MAY BE ADDED LATER]
C
      REAL*8 X0,X1,TV,FX1,SQPI,XL,XR
      PARAMETER (SQPI=1.7724538509055160D0)
      COMPLEX K2(10),KS1,C4,ZA,ZAC4
      REAL Y(1),X(500,5),B(1),PRNT(5),SIG(10),H(10),DER(2),
     1 BSAVE(20),W2(200), APPRES(500),TAR(500,2)
      EXTERNAL HZLOOP,FCTCI
      COMMON/PASSER/TV,LATE
      COMMON/TCOM/T(500),VSAVE(500)
      COMMON/PASS/ZAC4,ANORM,CURI,DC,SIG,B0,BM,SIG1,EPS,ISTEP
      COMMON/FPASS/AA,TMIN,TMAX,T0,TM,DB,BMTEST,
     * M1,M21,M2,JSPLN,NN,IFIRST,IOPT
      COMMON/SPLN/XS(200),YS(200),AS(200),BS(200),CS(200),NS,ISPLN
      COMMON/MODEL/K2,KS1,H,Z,A,R,HMAX,M
      DATA DER/2*0.0/,XMU0/1.2566371E-6/
      IF(IN.GT.1.OR.M.EQ.1) GO TO 20
      DO 10 J=2,M
      IF(B(J).EQ.B(J-1)) CALL ERRMSG('SOME SIG(J)=SIG(J-1)',4,6,16)
10    CONTINUE
20    DO 30 J=1,5
30    PRNT(J)=X(IN,J)
      IF(IN.GT.1) GO TO 800
      IF(IDER.EQ.1) GO TO 8001
35    SIG1=B(1)
      HMAX=A
      IF(M.EQ.1) GO TO 45
      DO 40 J=1,M1
      H(J)=B(M+J)
40    SIG(J)=B(J)
      CALL MINMAX(H,M1,HMIN,HMAX)
45    SIG(M)=B(M)
      ANORM=A/HMAX
      TCON=6.28318531E-7*SIG1*AA
      IF(JSPLN.EQ.0) GO TO 49
C--GET PRE-SPLINED FREQ FUNCTION (0<NB<12 OPTION)
      NS=0
      TEM=B0/DB
      ISPLN=0
46    TEM=TEM*DB
      IF(TEM.GE.BMTEST) GO TO 47
      NS=NS+1
      IF(NS.GT.200) CALL ERRMSG('SPLINED NS>200 IN FCODE',3,6,16)
      XS(NS)=TEM
      YS(NS)=HZLOOP(TEM*TEM)
      GO TO 46
47    CALL SPLIN1(NS,0.0,XS,YS,AS,BS,CS,0,DER,T,W2)
      ISPLN=1
```

```
49    TO=.5*TMIN/TCON
      TM=TMAX/TCON
      LATE=0
      IF(IFIRST.EQ.1) IWARN=0
      NEW=1
      TRANSL=1.E30
      DO 70 I=1,NN
      T(I)=X(I,1)/TCON
C--GET TRANSIENT IMPULSE RESPONSE VIA LAGGED CONVOLUTION IN TIME.
          TRANS=.63661977*RFLAGS(0,HZLOOP,EPS,TO,TM,T(I),NEW)
          NEW=0
        IF(TRANS.GT.1.0) GO TO 71
C--IF CALC.TRANS TOO NOISY, THEN FORCE TRANS=TRANSL; THIS SHOULD NOT
C  OCCUR WITH THE USUAL TIME RANGE USED WITH MOST FIELD EQUIPMENT.
        IF(TRANS.LT.0.0.OR.TRANS.GT.TRANSL) THEN
          TRANS=TRANSL
          IF(IWARN.EQ.0) THEN
            IWARN=1
            CALL WARN('NOISE IN CALC. TRANS DETECTED.',0,6,16,*71)
          ENDIF
        ENDIF
71    TRANSL=TRANS
      VSAVE(I)=TRANS
70    CONTINUE
C--IF IOPT=1, THEN CONVERT COMPUTED "TRANS" TO "APPRES"
      IF(IOPT.EQ.1) THEN
C**GET APP.RES. (SEE C**END OF "IF(IOPT.EQ.1) THEN" BELOW)
      DO 68 I=1,NN
        TIME=TCON*T(I)
        TV=T(I)*VSAVE(I)
C--MUELLERS ITER USING FCTCI(X1)=0 FOR SOLUTION X1 IN (0,20.)
      CALL DFIND(.1D-20,100,20.D0,FCTCI,XL,XR,IER)
      IF (IER.GT.0) THEN
        TAR(I,1)=1./SIG1
        X1=XR
        TAR(I,2)=0.0
        GO TO 62
      ENDIF
      CALL DRTMI(X1,FX1,FCTCI,XL,XR,.1D-5,1000,IER)
      IF(IER.GT.0) THEN
        TAR(I,1)=1./SIG1
        X1=XL
        TAR(I,2)=0.0
        GO TO 62
      ENDIF
C--X1 FOUND, GET APPRES TAR(I,1)
      TAR(I,1)=(XMUO*AA)/(4.*TIME*X1**2)
      IF(TIME.GT.0.04) LATE=1
C--LOOK FOR 2ND X1 AND TAR(I,2), ETC.
62    CALL DFIND(0.0D0,25,X1-.01*X1,FCTCI,XL,XR,IER)
      IF(IER.GT.0) THEN
        CALL DFIND(X1+.01*X1,25,1.0D5,FCTCI,XL,XR,IER)
        IF(IER.GT.0) THEN
          TAR(I,2)=0.0
          GO TO 68
          ENDIF
        ENDIF
        CALL DRTMI(X1,FX1,FCTCI,XL,XR,.1D-4,1000,IER)
        IF(IER.GT.0) THEN
          TAR(I,2)=0.0
          GO TO 68
        ENDIF
        TAR(I,2)=(XMUO*AA)/(4.*TIME*X1**2)
68      CONTINUE
C--GET FINAL APPARENT RESISTIVITY APPRES(I) FROM TAR(I,2)
        TEM=1./SIG1
        IF(ABS(TAR(1,1)-TEM).LT.ABS(TAR(1,2)-TEM)) THEN
          APPRES(1)=TAR(1,1)
        ELSE
          APPRES(1)=TAR(1,2)
        ENDIF
        IF(APPRES(1).EQ.0.0) APPRES(1)=TEM
        DO I=2,NN
          J=I-1
          IF(ABS(TAR(I,1)-APPRES(J)).LT.ABS(TAR(I,2)-APPRES(J))) THEN
            APPRES(I)=TAR(I,1)
          ELSE
            APPRES(I)=TAR(I,2)
          ENDIF
          IF(APPRES(I).EQ.0.0) APPRES(I)=APPRES(J)
        ENDDO
        ENDIF
C**END OF "IF(IOPT.EQ.1) THEN" ABOVE FOR APP.RES.
        IF(ISTEP.EQ.1) THEN
C--GET STEP RESPONSE AS INTEGRAL OVER TIME OF IMPULSE RESPONSE.
          CALL INTEG1(NN,T,VSAVE,3.0)
        ENDIF
        IF(IDER.EQ.0) GO TO 600
        IFIRST=0
        DO 80 J=1,M21
80      BSAVE(J)=B(J)
C--GET PRE-SPLINED TRANSIENT (EITHER ISTEP=0 OR 1) -OR- APPRES
600     IF(IOPT.EQ.0) THEN
            F=B(M2)*VSAVE(IN)/SIG1
        ELSE
          F=B(M2)*APPRES(IN)
        ENDIF
        RETURN
800     IF(IDER.EQ.0) GO TO 600
C--IDER=1 EST.DER.OPTION
8001    IF(IFIRST.EQ.1) GO TO 35
        DO 802 J=1,M21
          IF(B(J).NE.BSAVE(J)) GO TO 35
802     CONTINUE
        GO TO 600
        END
        REAL*8 FUNCTION FCTCI(X)
C--FUNCTION FCTCI(X) FOR ZERO OF FCTCI(X1)=0 VIA CALL DRTMI
        IMPLICIT REAL*8 (A-H,O-Z)
        PARAMETER (SQPI=1.772453850905516D0,CON1=2.D0/SQPI)
```

```
      COMMON/PASSER/TV,LATE
      X2=X*X
      IF(LATE.EQ.0.OR.X.GE.0.1) THEN
      E=DEXP(-X2)
      FCTCI=3.D0*DERF(X)-2.D0*X2*TV-CON1*X*(3.D0+2.D0*X2)*E
      ELSE
        FCTCI=(((-2.D0*X2/33.D0+2.D0/9.D0)*X2-
     1 4.D0/7.D0)*X2+0.8D0)*X2*X-SQPI*TV
      ENDIF
      RETURN
      END
      SUBROUTINE MARQ_TRANS_HZLOOP_SUBZ(Y,X,B,PRNT,NPRNT,N,TITLE,IOUT)
C-- INITIALIZATION ROUTINE (CALLED ONCE)
C
C  SUBZ IS CALLED BY NLSOL AFTER THE DATA Y(I),X(I,5) ARE READ.
C  SUBZ CHECKS FOR DATA ERRORS, READS ADDITIONAL SINIT
C  PARAMETERS, AND LOADS SOME CONSTANTS IN COMMON STORAGE...
C
C--PARAMETERS--
C       Y,X,B,PRNT SAME AS IN SUBROUTINE FCODE.
C       NPRNT=  CONTROL PARAMETERS TO USE PRNT(NPRNT) ARRAY
C               NPRNT REPRESENTS THE NO. X(I,NPRNT) VALUES
C       N=      NO. OBSERVATIONS GIVEN IN Y(N),X(N,5)
C       TITLE=  ALPHA TITLE ARRAY READ IN BY PGM IMSLMQ.
C       IOUT=   1 IF UNIT 6 AND 16 PRINT FILES USED
C               0 IF ONLY UNIT 6 PRINT FILE USED.
C
      CHARACTER*9 OPT(0:1)
      COMPLEX K2(10),KS1,C4,ZA,ZAC4
      CHARACTER*80 TITLE
      REAL Y(1),X(500,5),B(1),PRNT(1),SIG(10),H(10)
      COMMON/PASS/ZAC4,ANORM,CURI,DC,SIG,B0,BM,SIG1,EPS,ISTEP
      COMMON/FPASS/AA,TMIN,TMAX,TO,TM,DB,BMTEST,
     & M1,M21,M2,JSPLN,NN,IFIRST,IOPT
      COMMON/SPLN/FILL(1000),NS,ISPLN
      COMMON/MODEL/K2,KS1,H,Z,A,R,HMAX,M
C**      NAMELIST/INIT/MM,A,Z,EPS,B0,BM,NB
      COMMON/NAME_LIST/FILLS(65),MM,FILLS2(4),EPS_,
     1 FILLER(3031),IOPT_,ISTEP_,NB,B0_,PARM(4),BM_,A_,Z_
      DATA ISUBZ/0/,OPT/'TRANSIENT','APPRES'/
      IF(ISUBZ.NE.0) GO TO 10
C--PRESET
      ISUBZ=1
      MM=1
      ISTEP_=0
      A_=0.0
      Z_=0.0
      B0_=.01
      BM_=100.
      NB=8
      EPS_=.1E-9
      IOPT_=0
C**10    READ(5,INIT)
10    CALL NAMELIST(5,'SINIT',*11)
      IOPT=IOPT_
      M=MM
      EPS=EPS_
      ISTEP=ISTEP_
      B0=B0_
      BM=BM_
      A=A_
      Z=Z_
11    CALL NONBLANK(TITLE,NONBLK)
      WRITE(6,20) TITLE
20    FORMAT('1{NLSTCI}:',5X,A<NONBLK>/)
      IF(IOUT.EQ.1) WRITE(16,20) TITLE
      WRITE(6,30) MM,A,EPS,B0,BM,NB,Z,ISTEP,IOPT
      IF(IOUT.EQ.1) WRITE(16,30) MM,A,EPS,B0,BM,NB,Z,ISTEP,IOPT
30    FORMAT(' MM=',I3,12X,' A=',E13.6,4X,'EPS=',E13.6/
     & ' B0=',E13.6,2X,'BM=',E13.6,4X,'NB=',I3/
     & ' Z=',E13.6,3X,'ISTEP=',I3/' IOPT=',I3)
C--TEST $INIT PARMS
      IF(MM.LT.1.OR.MM.GT.10.OR.A.LE.0.0.OR.NB.LT.0.OR.
     & ISTEP.LT.0.OR.ISTEP.GT.1.OR.IOPT.LT.0.OR.IOPT.GT.1.OR.
     & BM.LE.B0.OR.B0.LE.0.0.OR.Z.LT.0.0)
     & CALL ERRMSG('SOME $INIT PARMS OUT OF RANGE.',6,6,16)
      IF(Z.GT.0.0.AND.ISTEP.EQ.1)
     1 CALL ERRMSG('Z>0 AND ISTEP=1 NOT ALLOWED.',1,6,16)
      IF(IOPT.EQ.1.AND.ISTEP.EQ.1)
     1 CALL ERRMSG('IOPT=1 AND ISTEP=1 NOT ALLOWED.',0,6,16)
C--TEST X(I, ) DATA BEFORE PROCEEDING
      DO 40 I=2,N
      IF(X(I,1).LE.X(I-1,1).OR.X(I,1).LE.0.0)
     & CALL ERRMSG('SOME X(I,1)<=0.0 OR NOT INCREASING.',7,6,16)
40    CONTINUE
C--PRESET SOME GLOBAL CONSTANTS
      IFIRST=1
      NN=N
      AA=A*A
      ZA=CMPLX(A,0.0)
      CURI=.3183098861/AA
      C4=CMPLX(A/(2.0*SQRT(AA+Z*Z)**3),0.0)
      ZAC4=ZA*C4
      DC=A*CURI*REAL(C4)
      TMIN=X(1,1)
      TMAX=X(N,1)
      ISPLN=0
      JSPLN=0
      IF(NB.GT.0.AND.NB.LT.12) JSPLN=1
      IF(JSPLN.EQ.1) THEN
       DB=EXP(2.30258509/FLOAT(NB))
       BMTEST=0.5*(BM+BM*DB)
      ENDIF
      WRITE(6,50)
      IF(IOUT.EQ.1) WRITE(16,50)
50    FORMAT(////' PARAMETER ORDER--'/)
      M1=MM-1
      M21=2*MM-1
      M2=M21+1
      WRITE(6,110) (I,I,I=1,MM)
      IF(IOUT.EQ.1) WRITE(16,110) (I,I,I=1,MM)
110   FORMAT(5X,I3,6X,6HSIGMA(,I3,1H))
      IF(MM.EQ.1) GO TO 132
      DO 120 I=1,M1
      J=MM+I
      IF(IOUT.EQ.1) WRITE(16,130) J,I
120   WRITE(6,130) J,I
130   FORMAT(5X,I3,6X,6HTHICK(,I3,1H))
132   WRITE(6,131) M2,M2,OPT(IOPT)
131   FORMAT(5X,I3,10X,'B(',I3,') SHIFT PARAMETER IN B(2*MM)*',A)
      IF(IOUT.EQ.1) WRITE(16,131) M2,M2,OPT(IOPT)
      NPRNT=2
      RETURN
      END
      SUBROUTINE NAMELIST(IUNIT,NAME,*)
C
C {NAMELIST INPUT ON VAX-11/780} VIA "CALL NAMELIST" (VERSION: 12/10/80)
C
C--A SIMULATED 'NAMELIST/NAME/' PROCESSOR FOR VAX-11 FORTRAN-77 TO
C  IMPLEMENT "CALL NAMELIST(IUNIT,'$NAME',*EOF)" ON VAX, WHICH
C  IS SIMILAR TO "READ(IUNIT,NAME,END=EOF)" ON MOST LARGE SYSTEMS.
C
C--BY W.L.ANDERSON, U.S. GEOLOGICAL SURVEY, DENVER, COLORADO.
C
C--THIS IS A SUBSET OF THE ACTUAL NAMELIST/NAME/ AVAILABLE ON
C  MOST LARGE MAIN-FRAME SYSTEMS. CURRENT OPTIONS ARE:
C
C  (1) ALL VARNAM'S ARE RESTRICTED TO 1 TO 6 CHAR'S (ALP,NUM, AND '_')
C      BUT MUST BEGIN WITH AN ALP CHAR (E.G., A3_, BVAR, C_2, ETC.)
C  (2) ONLY VARIABLE TYPES REAL*4 *8 (NAMTYP=1) AND INTEGER*2 *4
C      (NAMTYP=0). SEE C==== EXAMPLE STATEMENTS FOR NAMTYP BELOW =====.
C      {NOTE: COMPLEX,LOGICAL, OR CHARACTER VARIABLE TYPES ARE "NOT"
C      CODED IN THIS VERSION.}
C  (3) MAX. 60 VARNAM'S ALLOWED IN NAMELIST (FOR ALL '$NAMES' USED).
C  (4) MAX. NUMBER FIELD (FLOAT OR FIXED) IS 20 CHAR WIDE, WHERE
C      BLANK CHAR'S ARE IGNORED, AND TYPE CONVERSION IS AUTOMATIC.
C      FLOAT NUMBERS WITH OPTIONAL E+XX OR D-XX AND WITH OR WITHOUT '.'
C      IN THE MANTISSA IS ALLOWED (E.G., 123E-3, .123D+02, -3.14, ETC.).
C  (5) PARTIAL ARRAY'S ALLOWED; E.G., A(10)=25.1,
C      AND B=1,3.2,...
C  (6) REPEAT FACTORS ALLOWED; E.G., C=2*1,3,..
C  (7) ONLY 1-DIM ARRAYS ALLOWED WITH MAX SIZE 99999.
C  (8) THE NAMELIST '$NAME' MUST BE 2 TO 7 CHAR'S, AND MUST BEGIN WITH
C      A "$" CHAR (E.G., '$P', '$PARMS', ETC.); ALSO, THE FIRST CHAR IN
C      IFILE MAY BEGIN IN COL. 1 BUT LESS THAN COL. 72 (BUFFER IS 80).
C      LINES IN IFILE MAY BE CONTINUED TO COL. 1 ON NEXT LINE, AND
C      TERMINATE THE NAMELIST BY "$[END]"--THE "END" IS OPTIONAL. E.G.,
C
C      $PARMS A=1,B=2.3,7*1,C(3)=-.123E-10,
C      D=1800, E=5*20$END
C      $NEXNAM F=123, G=-10,C(2)=15.02 $
C      ...END-OF-IFILE...
C  (9) ABOUT 98% OF ALL THE POSSIBLE ERRORS ARE DETECTED AND AN
C      ERROR MESSAGE IS PRINTED ON UNIT 06, FOLLOWED BY CALL EXIT.
C      {NOTE: WATCH OUT FOR THE REMAINING 2% UNDETECTED ERRORS!}
```

```
C
C--SUBROUTINES CALLED:
C
C  DECODEIX, DECODEX, AND NONBLANK.
C
C--USAGE:
C
C  1. MODIFY FILE 'INCLNAMES.FOR' AS REQUIRED (USE ANY EDITOR).
C     (SEE C==== EXAMPLE STATEMENTS BELOW =====.)
C  2. RECOMPILE SUBROUTINE 'NAMELIST' WITH THE DESIRED INCLNAMES.FOR.
C  3. IN USERS CALLING PROGRAM, USE:
C     CALL NAMELIST(IUNIT,'$NAME',*N)  --ON VAX, WHERE N=E.O.F RETURN
C     STATEMENT LABEL.  THIS SIMULATES ON VAX:
C      'READ(IUNIT,NAME,END=N)' ON SYSTEMS WITH NAMELIST/NAME/...
C
C***********************************************************************
C
      CHARACTER*(*) NAME
      CHARACTER*1 C(47),BUFI
      CHARACTER*6 VARNAM
      CHARACTER*20 NUMFLD
      CHARACTER*80 BUF
C
C=======================================================================
C====== THE USER MUST CHANGE THE FOLLOWING STATEMENTS FOR THE SPECIFIC
C====== NAMELIST VARIABLES DESIRED (E.G., USE TECO OR EDT, ETC.)========
C====== DIMENSION NO_NAM VARIABLES TO AGREE WITH CHANGED DATA STATEMENTS
C==
C==ON VAX USE THE FOLLOWING INCLUDE STATEMENT (OPTIONALLY, USE /LIST):
C==
C>>    INCLUDE 'INCLNAMES.FOR/NOLIST'
C
C========================= INCLNAM13.FT ================================
C==================== FOR USE IN CALL NAMELIST =========================
C NORMALLY, ONE SHOULD COPY 'INCLNAM13.FT' TO 'INCLNAMES.FT'; THEN
C   EDIT 'INCLNAMES.FT' AS DESIRED FOR USERS CALL NAMELIST. NOTE THAT
C   ONE MUST RECOMPILE 'NAMELIST.FT' WITH USERS CALLING PROGRAM,
C   WHERE 'NAMELIST.FT' CONTAINS THE FOLLOWING STATEMENT:
C
C      INCLUDE 'INCLNAMES.FT/LIST'
C=======================================================================
C
C***********************************************************************
C  THIS IS "$PARMS AND $INIT" INPUT FOR PROGRAMS "NLSTCI" AND "NLSTCO"
C***********************************************************************
C
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$ CHANGE THE FOLLOWING FORTRAN-77 PARAMETER STATEMENT ONLY IF
C$$ INCREASING THE DEFAULT DIMENSIONS FOR NLSOL:
       PARAMETER (NDIM=500,MDIM=5,KDIM=20)
C$$ WHERE NDIM=MAX.OBS., MDIM=MAX.INDEP.VARS., KDIM=MAX.UNKNOWN PARMS.
C$$ DO NOT CHANGE THE FOLLOWING RELATED PARAMETER STATEMENT:
       PARAMETER (K1DIM=KDIM-1,
     1 IVDIM=KDIM+60,NKVDIM=96+2*NDIM+(KDIM*(7*KDIM+41))/2)
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
C
      COMMON/NAME_LIST/V1,V2,V3,V4,V5,V6,V7,V8,V9,V10,
     * V11,V12,V13,V14,V15,V16,V17,V18,V19,V20,
     * V21,V22,V23,V24,V25,V26,V27,V28,V29,V30,
     * V31,V32,V33,V34,V35,V36,V37,V38,V39,
     * V40,V41,V42,V43,V44,V45,V46,V47,V48,V49,V50,V51
      INTEGER V1,V2,V3,V4,V5,V6,V7,V8,V9,V10,V11,
     * V17, V21,V22,V23,V24,V25, V27,V28,V29, V35,V36,V37,V38,V39,
     * V40,V44,V45,V46
      DIMENSION V1(1),V2(1),V3(1),V4(1),
     * V5(1),V6(1),V7(1),V8(1),V9(1),V10(1),
     * V11(1),V12(1),V13(1),V14(1),V15(1),
     * V16(1),V17(1),V18(1),V19(1),V20(1),
     * V21(1),V22(1),V23(1),V24(1),V25(1),
     * V26(KDIM),V27(K1DIM),V28(1),V29(1),V30(1),
     * V31(1),V32(1),V33(1),V34(1),V35(1),
     * V36(1),V37(1),V38(1),V39(1),V40(IVDIM),
     * V41(NKVDIM),V42(KDIM),V43(KDIM),V44(1),V45(1),
     * V46(1),V47(1),V48(4),V49(1),V50(2),
     * V51(1),V52(1),V53(1),V54(1),V55(1),
     * V56(1),V57(1),V58(1),V59(1),V60(1)
      DIMENSION NAMDIM(60),NAMLEN(60),NAMTYP(60)
      CHARACTER*6 NAM(60)
      DATA NAM/'N','K','IP','M','IALT','ISTOP','IWT','IDER',
     * 'IPRT','NITER','INON','FF','T','E','TAU','XL','MODLAM',
     * 'GAMCR','DEL','ZETA','IOUT','SP','SCALEP','SY','SCALEY',
     * 'B','IB','IOB','MM','X0','Y0','L','EP','EPS','NEPS',
     * 'METHOD','NFIN','IER','MEV','IV','V','BL','BH',
     * 'IOPT','ISTEP','NB','B0','PARM','BM','A','Z',9*' '/
      DATA NAMDIM/25*1,KDIM,K1DIM,12*1,IVDIM,NKVDIM,2*KDIM,4*1,
     1 4,3*1,9*0/
      DATA NAMLEN/2*1,2,1,4,5,3,2*4,5,4,2,2*1,3,2,6,5,3,2*4,
     * 2,6,2,6,1,2,3,3*2,1,2,3,4,6,4,2*3,2,1,2*2,
     * 4,5,2*2,4,2,2*1,9*0/
      DATA NAMTYP/11*0,5*1,0,3*1,5*0,1,3*0,5*1,5*0,0,3*1,3*0,5*1,9*0/
      DATA NO_NAM/51/
C====== END OF INCLUDE STATEMENTS ====================================
C
C==
C== FOR EXAMPLE, FILE 'INCLNAMES.FOR' MAY CONTAIN (WITHOUT "C=="):
C==
C==       COMMON/NAME_LIST/V1,V2,V3,V4
C==       REAL*8 V1
C==       INTEGER V3
C==       DIMENSION V1(1),V2(2),V3(3),V4(4),
C==      * V5(1),V6(1),V7(1),V8(1),V9(1),V10(1),
C==      * V11(1),V12(1),V13(1),V14(1),V15(1),
C==      * V16(1),V17(1),V18(1),V19(1),V20(1),
C==      * V21(1),V22(1),V23(1),V24(1),V25(1),
C==      * V26(1),V27(1),V28(1),V29(1),V30(1),
C==      * V31(1),V32(1),V33(1),V34(1),V35(1),
C==      * V36(1),V37(1),V38(1),V39(1),V40(1),
C==      * V41(1),V42(1),V43(1),V44(1),V45(1),
C==      * V46(1),V47(1),V48(1),V49(1),V50(1),
C==      * V51(1),V52(1),V53(1),V54(1),V55(1),
C==      * V56(1),V57(1),V58(1),V59(1),V60(1)
C==       DIMENSION NAMDIM(60),NAMLEN(60),NAMTYP(60)
C==       CHARACTER*6 NAM(60)
C==       DATA NAM/'A','BB','ICC','DDD_4',56*' '/
C==       DATA NAMDIM/1,2,3,4,56*0/
C==       DATA NAMLEN/1,2,3,5,56*0/
C==       DATA NAMTYP/2*1,0,1,56*0/
C==       DATA NO_NAM/4/
C====== END OF EXAMPLE INCLUDE STATEMENTS ============================
C
C**********************************************************************
C NOTE: THE ABOVE EXAMPLE SIMULATES
C       'NAMELIST/NAME/A,BB,ICC,DDD_4'
C       'READ(IUNIT,NAME,END=EOF)'
C       'READ(IUNIT,ANYNAM,END=EOF)'
C       IN THE CALLING PROGRAM USING:
C       ...
C       REAL*8 A
C       ...
C       COMMON/NAME_LIST/A,BB(2),ICC(3),DDD_4(4)
C       ...
C       CALL NAMELIST(IUNIT,'$NAME',*EOF)
C       ...
C       CALL NAMELIST(IUNIT,'$ANYNAM',*EOF)
C       ...
C**********************************************************************
C
      DATA C/'A','B','C','D','E','F','G','H','I','J','K','L','M','N',
     * 'O','P','Q','R','S','T','U','V','W','X','Y','Z','_',
     * '1','2','3','4','5','6','7','8','9','0',
     * ' ','$','=',',','(','*',')','.','+','-'/
      J=LEN(NAME)
      IF(J.LT.2.OR.J.GT.7) THEN
        CALL ERRMSG('CALL NAMELIST ILLEGAL WITH NAME= '//
     1 NAME//'   (LENGTH<2 OR >7 CHAR''S)',1,6,0)
      ENDIF
      IF(NAME(1:1).NE.'$')
     1 CALL ERRMSG('CALL NAMELIST ILLEGAL WITH NAME= '//
     2 NAME//'   (1ST CHAR MUST BE "$" CHAR)',1,6,0)
C--INITIALIZE
      INAME=0
10    READ(IUNIT,11,END=99991,ERR=99992) BUF
11    FORMAT(A80)
      IF(INAME.EQ.1) GO TO 20
C--LOOK FOR "$NAME"
        I=INDEX(BUF,NAME)
      IF(I.EQ.0) GO TO 10
      INAME=1
      ICOL=I+J
      JNAM=0
      ILEN=0
      VARNAM=' '
      NUMLEN=0
      IELE=1
      GO TO 30
```

```
20    ICOL=1
30    CALL NONBLANK(BUF,LENBUF)
C==BEGIN PARSER LOOP (THE BIG 20000 LOOP)
      IEND=0
      DO 20000 I=ICOL,LENBUF
         BUFI=BUF(I:I)
      DO 40 IC=1,27
         IF(BUFI.EQ.C(IC)) GO TO 100
40    CONTINUE
      DO 50 IC=28,37
      IF(BUFI.EQ.C(IC)) GO TO 200
50    CONTINUE
      DO 60 IC=38,47
      IC_=IC-37
      IF(BUFI.EQ.C(IC)) GO TO 70
60    CONTINUE
61    WRITE(6,66) I,BUF
66    FORMAT(/' (NAMELIST): ERROR IN FOLLOWING RECORD AT COL(',I2,'):'/
     1 1X,A80/<I>X,'*')
      CALL ERRMSG('ILLEGAL CHAR="'//BUFI//'" FOUND',0,6,0)
67    WRITE(6,66) I,BUF
      CALL ERRMSG('NUMLEN<1 IN DECODEIX    ',0,6,0)
68    WRITE(6,66) I,BUF
      CALL ERRMSG('NUMLEN<1 IN DECODEX',0,6,0)
70    GO TO (20000,72,73,74,75,76,77,78,79,79),IC_
C--'$' CHAR
72    IEND=1
      IF(NUMLEN.GT.0) GO TO 798
      IF(JNAM.EQ.0) GO TO 99990
      WRITE(6,66) I,BUF
      CALL ERRMSG('MISPLACED "$" CHAR',0,6,0)
C--'=' CHAR
73    IEQ=1
C--CHECK FOR VALID VARNAM, LENGTH ILEN, ETC.
      IF(ILEN.LT.1) GO TO 733
      DO 732 J=1,NO_NAM
      JNAM=J
      JLEN=NAMLEN(J)
      IF(JLEN.NE.ILEN) GO TO 732
      DO 731 K=1,JLEN
      IF(VARNAM(K:K).NE.NAM(JNAM)(K:K)) GO TO 732
731   CONTINUE
C--VARNAM VERIFIED OK TO PROCEED TO NUMFLD(S)
C
      IDIM=NAMDIM(JNAM)
      NUMLEN=0
      NDEC=0
      NREP=1
      NEXP=0
      GO TO 20000
732   CONTINUE
      WRITE(6,66) I,BUF
      CALL ERRMSG('ILLEGAL VARNAM='//VARNAM//' FOUND',0,6,0)
733   WRITE(6,66) I,BUF
      CALL ERRMSG('MISPLACED "=" CHAR  ',0,6,0)
```

```
C--',' CHAR
74     IF(NUMLEN.GT.0) GO TO 799
       WRITE(6,66) I,BUF
       CALL ERRMSG('MISPLACED "," CHAR',0,6,0)
C--'(' CHAR
75     IELE=0
       GO TO 20000
C--'*' CHAR
76     IF(JNAM.EQ.0.OR.NUMLEN.LT.1.OR.NUMLEN.GT.5) GO TO 767
760    CALL DECODEIX(NUMFLD,NUMLEN,NREP,*67)
       NUMLEN=0
       IF(NREP.GT.0.AND.NREP.LE.NAMDIM(JNAM)) GO TO 20000
       WRITE(6,66) I,BUF
       CALL ERRMSG('REPEAT FACTOR <1 OR >NAMDIM   ',0,6,0)
767    WRITE(6,66) I,BUF
       CALL ERRMSG('REPEAT WIDTH > 5 OR MISPLACED "*" CHAR',0,6,0)
C--')' CHAR
77     IF(IELE.NE.0) GO TO 772
       CALL DECODEIX(NUMFLD,NUMLEN,IELE,*67)
       IF(IELE.LT.1) GO TO 773
       NREP=1
       GO TO 20000
772    WRITE(6,66) I,BUF
       CALL ERRMSG('MISPLACED ")" CHAR',0,6,0)
773    WRITE(6,66) I,BUF
       CALL ERRMSG('ARRAY IELE<1 OR >NAMDIM  ',0,6,0)
C--'.' CHAR
78     IF(JNAM.EQ.0.OR.NEXP.GT.0.OR.NDEC.GT.0) GO TO 781
       NDEC=NUMLEN+1
       IF(NAMTYP(JNAM).EQ.1) GO TO 200
781    WRITE(6,66) I,BUF
       CALL ERRMSG('MISPLACED "." CHAR',0,6,0)
C--'-' OR '+' CHAR
79     IF(IELE.GT.0.OR.NEXP.GT.0) GO TO 210
       WRITE(6,66) I,BUF
       CALL ERRMSG('MISPLACED "-" OR "+" CHAR',0,6,0)
C--<ALP> CHAR
100    IF(NUMLEN.GT.0) GO TO 209
       IF(ILEN.GT.0) GO TO 102
       IEQ=0
       IELE=1
102    ILEN=ILEN+1
       IF(ILEN.GT.6) GO TO 101
       VARNAM(ILEN:ILEN)=BUFI
       GO TO 20000
101    WRITE(6,66) I,BUF
       CALL ERRMSG('VARNAM>6 CHAR''S',0,6,0)
C--<+-NUM> CHAR
200    IF(IELE.EQ.0) GO TO 210
       IF(IEQ.EQ.0) GO TO 102
       GO TO 210
209    IF(BUFI.EQ.'E'.OR.BUFI.EQ.'D') THEN
         NEXP=NUMLEN+1
       ELSE
         GO TO 61
```

```
      ENDIF
210   NUMLEN=NUMLEN+1
      IF(NUMLEN.GT.20) GO TO 211
      NUMFLD(NUMLEN:NUMLEN)=BUFI
      GO TO 20000
211   WRITE(6,66) I,BUF
      CALL ERRMSG('NUM FIELD>20 CHAR''S',0,6,0)
C--PROCESS NUMBER FIELD
799   IDIM=IDIM-1
      IF(IDIM.LT.0) GO TO 10004
798   IF(NEXP.GT.0) GO TO 1000
C--[NEXP=0]
      IF(NDEC.GT.0) GO TO 899
C--[NEXP=0, NDEC=0]
      CALL DECODEIX(NUMFLD,NUMLEN,IX,*67)
C--CONVERT IX AND STORE IN COMMON
800   X=IX
      IF(IELE.GT.NAMDIM(JNAM)) GO TO 773
8000  GO TO (801,802,803,804,805,806,807,808,809,810,
     * 811,812,813,814,815,816,817,818,819,820,
     * 821,822,823,824,825,826,827,828,829,830,
     * 831,832,833,834,835,836,837,838,839,840,
     * 841,842,843,844,845,846,847,848,849,850,
     * 851,852,853,854,855,856,857,858,859,860),JNAM
801   V1(IELE)=X
      GO TO 10000
802   V2(IELE)=X
      GO TO 10000
803   V3(IELE)=X
      GO TO 10000
804   V4(IELE)=X
      GO TO 10000
805   V5(IELE)=X
      GO TO 10000
806   V6(IELE)=X
      GO TO 10000
807   V7(IELE)=X
      GO TO 10000
808   V8(IELE)=X
      GO TO 10000
809   V9(IELE)=X
      GO TO 10000
810   V10(IELE)=X
      GO TO 10000
811   V11(IELE)=X
      GO TO 10000
812   V12(IELE)=X
      GO TO 10000
813   V13(IELE)=X
      GO TO 10000
814   V14(IELE)=X
      GO TO 10000
815   V15(IELE)=X
      GO TO 10000
816   V16(IELE)=X
```

```
      GO TO 10000
817   V17(IELE)=X
      GO TO 10000
818   V18(IELE)=X
      GO TO 10000
819   V19(IELE)=X
      GO TO 10000
820   V20(IELE)=X
      GO TO 10000
821   V21(IELE)=X
      GO TO 10000
822   V22(IELE)=X
      GO TO 10000
823   V23(IELE)=X
      GO TO 10000
824   V24(IELE)=X
      GO TO 10000
825   V25(IELE)=X
      GO TO 10000
826   V26(IELE)=X
      GO TO 10000
827   V27(IELE)=X
      GO TO 10000
828   V28(IELE)=X
      GO TO 10000
829   V29(IELE)=X
      GO TO 10000
830   V30(IELE)=X
      GO TO 10000
831   V31(IELE)=X
      GO TO 10000
832   V32(IELE)=X
      GO TO 10000
833   V33(IELE)=X
      GO TO 10000
834   V34(IELE)=X
      GO TO 10000
835   V35(IELE)=X
      GO TO 10000
836   V36(IELE)=X
      GO TO 10000
837   V37(IELE)=X
      GO TO 10000
838   V38(IELE)=X
      GO TO 10000
839   V39(IELE)=X
      GO TO 10000
840   V40(IELE)=X
      GO TO 10000
841   V41(IELE)=X
      GO TO 10000
842   V42(IELE)=X
      GO TO 10000
843   V43(IELE)=X
      GO TO 10000
```

```
844   V44(IELE)=X
      GO TO 10000
845   V45(IELE)=X
      GO TO 10000
846   V46(IELE)=X
      GO TO 10000
847   V47(IELE)=X
      GO TO 10000
848   V48(IELE)=X
      GO TO 10000
849   V49(IELE)=X
      GO TO 10000
850   V50(IELE)=X
      GO TO 10000
851   V51(IELE)=X
      GO TO 10000
852   V52(IELE)=X
      GO TO 10000
853   V53(IELE)=X
      GO TO 10000
854   V54(IELE)=X
      GO TO 10000
855   V55(IELE)=X
      GO TO 10000
856   V56(IELE)=X
      GO TO 10000
857   V57(IELE)=X
      GO TO 10000
858   V58(IELE)=X
      GO TO 10000
859   V59(IELE)=X
      GO TO 10000
860   V60(IELE)=X
      GO TO 10000
C--[NEXP=0, NDEC>0]
899   CALL DECODEX(NUMFLD,NUMLEN,NDEC,X,*68)
C--CONVERT X AND STORE IN COMMON
900   IF(IELE.GT.NAMDIM(JNAM)) GO TO 773
      GO TO 8000
C--[NEXP>0]
1000  IF(NDEC.GT.0) GO TO 2000
C--[NEXP>0, NDEC=0]
      CALL DECODEIX(NUMFLD,NEXP-1,IX,*67)
      X=IX
1002  J=1
      DO 1001 K=NEXP+1,NUMLEN
      NUMFLD(J:J)=NUMFLD(K:K)
1001  J=J+1
      CALL DECODEIX(NUMFLD,NUMLEN-NEXP,IE,*67)
      X=X*10.**IE
C** (LATER INSERT A CALL TO A OVERFLOW HANDLER, ETC.)
      GO TO 900
C--[NEXP>0, NDEC>0]
2000  CALL DECODEX(NUMFLD,NEXP-1,NDEC,X,*68)
      GO TO 1002
```

```
C--NEXT IELE?
10000 IELE=IELE+1
      IF(IELE.GT.NAMDIM(JNAM)) GO TO 10002
      IF(NREP.GT.1) GO TO 10003
10001 IF(IEND.EQ.1) GO TO 99990
      NUMLEN=0
      NDEC=0
      NEXP=0
      NREP=1
      ILEN=0
      VARNAM=' '
      GO TO 20000
10002 IELE=1
      GO TO 10001
10003 NREP=NREP-1
      IDIM=IDIM-1
      IF(IDIM.GE.0) GO TO 8000
10004 WRITE(6,66) I,BUF
      CALL ERRMSG('TOO MANY ELEMENTS FOR GIVEN NAMDIM.',0,6,0)
C==END OF DO 20000   CONTINUE PARSER -OR- READ IN NEXT BUF, ETC.
20000 CONTINUE
                GO TO 10
C--'S' CHAR (DELIMITER S[END] FOR THIS SNAME --S)
99990 RETURN
C--E.O.F. ON FILE IUNIT ENCOUNTERED.
99991 RETURN 1
99992 CALL ERRMSG('CANNOT OPEN/READ CALL NAMELIST(IFILE,...)',1,6,0)
      END
      SUBROUTINE DUMYPCODE()
C--DUMMY PCODE FOR USE IN 'MARQRT' OR 'NLSOL'
      CALL ERRMSG('IDER=0 NOT AVAILABLE IN THIS VERSION.',4,6,16)
      END
      SUBROUTINE SIGSUBEND(Y,X,B,K,N,TITLE,IOUT)
C**GENERAL SUBEND TERMINATION ROUTINE WITH 'SIGMA' NAMES.
C  ALSO GIVES RESTART SPARMS ON UNIT=4 AS 'FOR005.TMP'
C
      CHARACTER*132 LINE
      CHARACTER*80 TITLE
      REAL Y(1),X(500,5),B(1)
      CALL NONBLANK(TITLE,NB)
      WRITE(6,10) TITLE
10    FORMAT(//' ********  E N D  ********',5X,A<NB>//
     1 ' PARAMETER NAME',6X,'FINAL SOLUTION',8X,
     2 'RESISTIVITY   LAYER DEPTH'/)
      IF(IOUT.EQ.1) WRITE(16,10) TITLE
      MM=(K+1)/2
      DO 30 I=1,MM
      R=1.0/B(I)
      WRITE(6,20) I,I,B(I),I,R
20    FORMAT(2X,I3,3X,'SIGMA(',I2,') =',E16.8,2X,I2,E16.8)
      IF(IOUT.EQ.1) WRITE(16,20) I,I,B(I),I,R
30    CONTINUE
      K1=0
      IF(K.EQ.1) GO TO 60
      IF(K.EQ.2) GO TO 52
      M2=MM+1
      K1=K
      IF(MOD(K,2).EQ.0) K1=K-1
      D=0.0
      DO 50 I=M2,K1
      D=D+B(I)
      L=I-MM
      WRITE(6,40) I,L,B(I),L,D
40    FORMAT(2X,I3,3X,'THICK(',I2,') =',E16.8,22X,I2,E16.8)
      IF(IOUT.EQ.1) WRITE(16,40) I,L,B(I),L,D
50    CONTINUE
      IF(K1.EQ.K) GO TO 60
52    WRITE(6,54) K,B(K)
54    FORMAT(2X,I3,3X,'SHIFT',5X,'=',E16.8)
      IF(IOUT.EQ.1) WRITE(16,54) K,B(K)
C** GENERATE RESTART SPARMS ON FOR005.TMP
60    REWIND 5
      OPEN(UNIT=4,FILE='FOR005.TMP',STATUS='NEW',
     1 CARRIAGECONTROL='LIST')
      READ(5,65,END=999) LINE
65    FORMAT(A)
      CALL NONBLANK(LINE,NB)
      WRITE(4,66) LINE
66    FORMAT(A<NB>)
      IDOL=0
70    READ(5,65,END=999) LINE
      I=INDEX(LINE,'$')
      IF(I.NE.0) THEN
        IF(IDOL.EQ.0) THEN
          IDOL=1
          J=INDEX(LINE(I+1:),'$')
          IF(J.NE.0) THEN
            IDOL=2
            LINE(J:J)=','
          ENDIF
        ELSE
          IDOL=2
          LINE(I:I)=','
        ENDIF
      ENDIF
      CALL NONBLANK(LINE,NB)
      WRITE(4,66) LINE
      IF(IDOL.LT.2) GO TO 70
      LINE(1:)='B='
      DO 80 I=1,K
      ENCODE(16,90,LINE(3:18)) B(I)
90    FORMAT(G16.8)
      IF(I.LT.K) THEN
        LINE(19:19)=','
      ELSE
        LINE(19:19)='$'
      ENDIF
      CALL NONBLANK(LINE,NB)
      WRITE(4,66) LINE
      LINE(1:2)='  '
```

```
80    CONTINUE
100   READ(5,65,END=999) LINE
      CALL NONBLANK(LINE,NB)
      WRITE(4,66) LINE
      GO TO 100
999   RETURN
      END
      SUBROUTINE CPUTIME(I1,I2)
C
C  CPUTIME WRITES "ELAPSED & CPU" TIME FROM PREVIOUS "CALL SETTIME" ON
C  FORTRAN UNITS I1 (IF NOT 0) AND I2 (IF NOT 0).
C
C  WILL EJECT FIRST IF I1>0 (OR I2>0).
C  DOUBLE SPACE FIRST IF I1<0 (OR I2<0).
C
C  E.G., USE TO TIME ELAPSED & CPU TIME FOR PROGRAM OR CODE SEGMENTS AS:
C
C     CALL SETTIME    ! DON'T FORGET TO DO THIS!
C     >>>>> THE CODE TO TIME IS HERE <<<<<  ! USUALLY A COMPLETE PROGRAM
C     CALL CPUTIME(-6,16) ! OR USE I1 OR I2=0 TO OMIT WRITE.
C
      SAVE
      INTEGER*4 ABSVAL(4),INCRVAL(4)
      CALL PROCINFO(ABSVAL,INCRVAL)
      TIMES=SECNDS(TIME0)
      MIN=TIMES/60.0
      SEC=AMOD(TIMES,60.0)
      CPUSEC=INCRVAL(1)*.01
      IMIN=CPUSEC/60.0
      CSEC=AMOD(CPUSEC,60.0)
      PCPU=100.*(CPUSEC/TIMES)
      IF(I1.NE.0) THEN
        IF(I1.GT.0) THEN
          J=1
        ELSE
          J=0
        ENDIF
      WRITE(IABS(I1),60) J,TIMES,MIN,SEC,CPUSEC,IMIN,CSEC,PCPU,
     1 (INCRVAL(I),I=2,4)
60    FORMAT(I1,65('$')/' TOTAL "ELAPSED" TIME=',F16.2,' SEC. (',
     1 I4,' MIN.',F6.2,' SEC.)'/
     2 ' CPU_TIME=',F15.2,' SEC. (',I4,' M. ',F5.2,
     1 ' S.)   CPU % =',F6.2,'%'/
     3 ' BUF.I/O_COUNT=',I10/
     4 ' DIR.I/O_COUNT=',I10/
     5 ' PAGE_FAULTS=',2X,I10/
     6 ' ',65('$')//)
      ENDIF
      IF(I2.NE.0) THEN
        IF(I2.GT.0) THEN
          J=1
        ELSE
          J=0
        ENDIF
      WRITE(IABS(I2),60) J,TIMES,MIN,SEC,CPUSEC,IMIN,CSEC,PCPU,
     1 (INCRVAL(I),I=2,4)
      ENDIF
      RETURN
C** ENTRY 'CALL SETTIME'--MUST BE DONE BEFORE 'CALL CPUTIME(I1,I2)'
      ENTRY SETTIME()
      TIME0=SECNDS(0.0)
      CALL PROCINFO(ABSVAL,INCRVAL)
      RETURN
      END
      SUBROUTINE DECODEIX(NUMFLD,NUMLEN,IX,*)
C--USED IN CALL NAMELIST(IUNIT,'SNAME',*)
      CHARACTER*9 FMT
      CHARACTER*20 NUMFLD
      IF(NUMLEN.LT.1) RETURN 1
      IDIFF=20-NUMLEN
      IF(IDIFF.EQ.0) THEN
        ENCODE(9,991,FMT) NUMLEN
      ELSE
        ENCODE(9,992,FMT) NUMLEN,IDIFF
      ENDIF
991   FORMAT('(I',I2,'    )')
992   FORMAT('(I',I2,',',I2,'X)')
      DECODE(9,FMT,NUMFLD) IX
      RETURN
      END
      SUBROUTINE DECODEX(NUMFLD,NUMLEN,NDEC,X,*)
C--USED IN CALL NAMELIST(IUNIT,'SNAME',*)
      CHARACTER*12 FMT
      CHARACTER*20 NUMFLD
      IF(NUMLEN.LT.1) RETURN 1
      LENDEC=NUMLEN-NDEC
      IDIFF=20-NUMLEN
      IF(IDIFF.EQ.0) THEN
       ENCODE(12,991,FMT) NUMLEN,LENDEC
      ELSE
       ENCODE(12,992,FMT) NUMLEN,LENDEC,IDIFF
      ENDIF
991   FORMAT('(F',I2,'.',I2,'    )')
992   FORMAT('(F',I2,'.',I2,',',I2,'X)')
      DECODE(12,FMT,NUMFLD) X
      RETURN
      END
      REAL*8 FUNCTION DERF(X)
C
C  DERF COMPUTES THE ERROR FUNCTION TO ABOUT 15-PLACES.
C  SEE MATH. OF COMP., V.22,N.101,JAN,1968.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A1(19),A2(19)
      DATA PI/3.141592653589793D0/
      DATA A1/.7032250027437754D0,.3305015219166062D0,
     1 .2013397472647063D0,.1086302450227407D0,
     2 .4677552343248486D-1,.1539857261571020D-1,
     3 .3801507679852987D-2,.6971837924080287D-3,
     4 .9449092688104550D-4,.9432811698383668D-5,
     5 .6919275203251401D-6,.3722523493691080D-7,
     6 .1466606142338001D-8,.4226161443180049D-10,
     7 .8897865267233D-12,.136760444757D-13,
     8 .1533423425D-15,.12536751D-17,.74517D-20/
      DATA A2/.2472551681400521D0,.1442272263615747D0,
     1 .8698945499593455D-1,.4397733819408337D-1,
     2 .1724396258866226D-1,.5079069612202570D-2,
     3 .1108606453423407D-2,.1782280162548617D-3,
     4 .2104045830732514D-4,.1820663163643408D-5,
     5 .1153309909443694D-6,.5342750276030827D-8,
     6 .1808485878095127D-9,.4469682292488lD-11,
     7 .806068838945D-13,.10601364636D-14,
     8 .101649277D-16,.710005D-19,0.0D0/
      IF(X.EQ.0.0D0) THEN
        DERF=0.0D0
        RETURN
      ENDIF
        B=.4D0*X
        S=DSIN(B)
        C=DCOS(B)
        C2=C+C
        ALP=C2*C-1.D0
        SUM=0.0D0
        DO 10 N=1,19
          SUM=SUM+(A1(N)+C2*A2(N))*ALP**(N-1)
10      CONTINUE
      DERF=B/PI+S*SUM
      RETURN
      END
      SUBROUTINE DFIND(XO,NHALF,XM,FCT,XL,XR,IER)
C
C--"FIND" FIRST FCT(XL) AND FCT(XR) WITH OPPOSITE SIGNS
C  IN RANGE X=(XO,XM] USING MAX. 2*NHALF BISECTIONS, WHERE
C  FCT IS A REAL*8 EXTERNAL DECLARED FUNCTION.
C
C--IF NO SIGN CHANGE IN (XO,XM],  EXITS WITH IER=1
C  (ELSE IER=0 MEANS XL AND XR FOUND).
C
C--USE BEFORE CALL DRTMI(X,F,FCT,XL,XR,EPS,MAXITR,IER)
C  TO FIND GUARANTEED ROOT OF FCT(X)=0 BY MUELLERS ITERATION.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      XL=XO
      SIGNXR=DSIGN(1.D0,FCT(XM))
      XR=XM
      DO N=1,NHALF
        X=0.5D0*(XL+XR)
        IF(DSIGN(1.D0,FCT(X)).NE.SIGNXR) GO TO 10
        XR=X
      ENDDO
      XL=0.5D0*(XO+XM)
      XR=XM
      DO N=1,NHALF
        X=0.5D0*(XL+XR)
        IF(DSIGN(1.D0,FCT(X)).NE.SIGNXR) GO TO 20
        XL=X
      ENDDO
      IER=1
      RETURN
10    IER=0
      XL=X
      RETURN
20    IER=0
      XR=X
      END
      SUBROUTINE DRTMI(X,F,FCT,XLI,XRI,EPS,IEND,IER)
C--IBM SSP ROUTINE, P.219, TO SOLVE FCT(X)=0 BY MUELLERS ITERATION
      IMPLICIT REAL*8 (A-H,O-Z)
      IER=0
      XL=XLI
      XR=XRI
      X=XL
      TOL=X
      F=FCT(TOL)
      IF(F)1,16,1
1     FL=F
      X=XR
      TOL=X
      F=FCT(TOL)
      IF(F)2,16,2
2     FR=F
      IF(DSIGN(1.D0,FL)+DSIGN(1.D0,FR))25,3,25
3     I=0
      TOLF=100.D0*EPS
4     I=I+1
      DO 13 K=1,IEND
      X=.5D0*(XL+XR)
      TOL=X
      F=FCT(TOL)
      IF(F)5,16,5
5     IF(DSIGN(1.D0,F)+DSIGN(1.D0,FR))7,6,7
6     TOL=XL
      XL=XR
      XR=TOL
      TOL=FL
      FL=FR
      FR=TOL
7     TOL=F-FL
      A=F*TOL
      A=A+A
      IF(A-FR*(FR-FL))8,9,9
8     IF(I-IEND)17,17,9
9     XR=X
      FR=F
      TOL=EPS
      A=DABS(XR)
      IF(A-1.D0)11,11,10
10    TOL=TOL*A
11    IF(DABS(XR-XL)-TOL)12,12,13
12    IF(DABS(FR-FL)-TOLF)14,14,13
```

```
13    CONTINUE
      IER=1
14    IF(DABS(FR)-DABS(FL))16,16,15
15    X=XL
      F=FL
16    RETURN
17    A=FR-F
      DX=(X-XL)*FL*(1.D0+F*(A-TOL)/(A*(FR-FL)))/TOL
      XM=X
      FM=F
      X=XL-DX
      TOL=X
      F=FCT(TOL)
      IF(F)18,16,18
18    TOL=EPS
      A=DABS(X)
      IF(A-1.D0)20,20,19
19    TOL=TOL*A
20    IF(DABS(DX)-TOL)21,21,22
21    IF(DABS(F)-TOLF)16,16,22
22    IF(DSIGN(1.D0,F)+DSIGN(1.D0,FL))24,23,24
23    XR=X
      FR=F
      GO TO 4
24    XL=X
      FL=F
      XR=XM
      FR=FM
      GO TO 4
25    IER=2
      RETURN
      END
      SUBROUTINE ERRMSG(MSG,ISKIP,IUNIT1,IUNIT2)
C
C  GENERAL ERROR MESSAGE OUTPUT AND EXIT ON VAX-11/780
C
C  MSG*(*) = VARIABLE-LENGTH 'MESSAGE'
C  ISKIP = 0 FOR NO BLANK LINE BEFORE OUTPUT TO IUNIT1 & IUNIT2
C        > 0 FOR ONE BLANK LINE BEFORE.
C  IUNIT1 = 0 TO SUPPRESS OUTPUT ON IUNIT1 (>0 TO WRITE ON IUNIT1).
C  IUNIT2 = 0 TO SUPPRESS OUTPUT ON IUNIT2 (>0 TO WRITE ON IUNIT2).
C
C  MESSAGES ARE WRITTEN IN THE FORM:
C
C  (ERRMSG): _MSG_HERE_
C
      CHARACTER*(*) MSG
      I=LEN(MSG)
      DO 1 J=1,2
        IF(J.EQ.1) THEN
       JUNIT=IUNIT1
          ELSE
       JUNIT=IUNIT2
        ENDIF
        IF(JUNIT.GT.0) THEN
```

```
          IF(ISKIP.EQ.0) THEN
            WRITE(JUNIT,2) MSG
            ELSE
       WRITE(JUNIT,3) MSG
          ENDIF
      ENDIF
1     CONTINUE
      CALL EXIT
2     FORMAT(1X,'{ERRMSG}: ',A<I>)
3     FORMAT(/1X,'{ERRMSG}: ',A<I>)
      END
      SUBROUTINE INTEG1(N,X,Y,Y0)
C        THIS ROUTINE INTEGRATES A FUNCTION"S VALUES (Y
C        AS A FUNCTION OF X) FROM 0 TO X BY CALCULATING THE CUBIC
C        SPLINE COEFFICIENTS AND INTEGRATING THE RESULTING
C        CUBIC POLYNOMIAL APPROXIMATION.  THE Y VALUES ARE
C        REPLACED BY THE INTEGRATED VALUES.
C        Y0 IS THE VALUE OF Y AT X=0.0 (ASSUMES THAT ALL INPUT
C        X > 0).
      DIMENSION X(N),Y(N)
      DIMENSION A(200),B(200),C(200),P(200),S(200),PS(2),X1(200),Y1(200)
      DATA PS/0.0,0.0/
      DO 1 I=1,N
      X1(I+1)=X(I)
    1 Y1(I+1)=Y(I)
      X1(1)=0.0
      Y1(1)=Y0
      N1=N+1
      CALL SPLIN1(N1,0,X1,Y1,A,B,C,0,PS,P,S)
      Y(1)=X(1)*(Y0+X(1)*A(1)/2.+X(1)*X(1)*B(1)/3.+X(1)**3*C(1)/4.)
      N1=N-1
      DO 10 I=1,N1
      Z=X(I+1)-X(I)
   10 Y(I+1)=Y(I)+Z*(Y1(I+1)+A(I+1)*Z/2.+B(I+1)*Z*Z/3.+C(I+1)*Z**3/4.)
      RETURN
      END
      SUBROUTINE MINMAX(A,N,AMIN,AMAX)
      DIMENSION A(1)
      AMIN=A(1)
      AMAX=AMIN
      DO 1 I=2,N
      AMIN=AMIN1(AMIN,A(I))
      AMAX=AMAX1(AMAX,A(I))
    1 CONTINUE
      RETURN
      END
      SUBROUTINE NLSOL(FCODE,PCODE,SUBZ,SUBEND)
C
C {NLSOL}: GENERAL NONLINEAR LEAST-SQUARES SOLUTION   {2/8/82}
C          USING DENNIS ET AL (1979; SEE REF1 BELOW)
C          ADAPTIVE NONLINEAR LEAST-SQUARES ALGORITHM.
C
C** THIS IS AN INTERFACE ROUTINE WRITTEN FOR THE VAX-11/780 BY
C    W.L.ANDERSON, U.S.GEOLOGICAL SURVEY, DENVER, COLORADO.
C
```

```
C** THIS INTERFACE (NLSOL) HAS ADDITIONAL OPTIONS (BESIDE REF1) TO:
C   (1) PERFORM EITHER UNCONSTRAINED OR UP TO 4-TYPES OF CONSTRAINED
C       ADAPTIVE NONLINEAR REGRESSION FOR ARBITRARY NONLINEAR PROBLEMS.
C       (I.E., PARTIAL OR FULL LOWER/HIGHER PARAMETER BOUNDS, ETC.)
C   (2) HOLDING CERTAIN PARAMETERS FIXED (I.E., AS CONSTANTS) IN THE
C       LEAST-SQUARES (THIS IS ANOTHER FORM OF CONSTRAINING SOLUTION
C       SPACE).
C   (3) PROVIDE FOR WEIGHTED OBSERVATIONS (I.E., WEIGHTED LEAST-SQUARES)
C   (4) OBJECT (RUN)-TIME CONTROL OF READING THE DATA MATRIX, PLUS
C       MANY OTHER I/O OPTIONS, ETC.
C   (5) OPTIONALLY, ONE CAN USE EITHER ESTIMATED PARTIAL DERIVATIVES, OR
C       ANALYTICAL PARTIAL DERIVATIVES (IF SUBROUTINE PCODE AVAILABLE).
C
C** THE USER ONLY NEEDS TO WRITE SUBROUTINES FCODE, PCODE, SUBZ, AND
C   SUBEND (SEE DETAILS BELOW) EXACTLY AS USED IN SUBROUTINE 'MARQRT'
C   (SEE REF2) OR 'IMSLMQ' (SEE REF3).  ALSO, THE SAME PARAMETER FILE
C   FOR005 AND OBJECT (RUN)-TIME DATA MATRIX FILE FOR010 AS USED BY
C   EITHER MARQRT OR IMSLMQ MAY BE USED IN 'NLSOL'.
C
C** NLSOL CALLS NLITR WHICH CALLS 'NL2ITR' AS PUBLISHED BY DENNIS ET AL,
C   (SEE REF1, P. 38), OR 'NL2SNO' (SEE REF1, P. 35).
C
C** REF1:  DENNIS, J.E., ET AL, 1979, AN ADAPTIVE NONLINEAR LEAST-
C          SQUARES ALGORITHM, NTIS REPORT AD-A079-716.
C
C   REF2:  ANDERSON, W.L., 1980, PROGRAM MARQHXY: INVERSION OF HX AND HY
C          FREQUENCY SOUNDINGS FROM A GROUNDED WIRE SOURCE, USGS OPEN-
C          FILE REPT. 80-901.
C
C   REF3:  ANDERSON, W.L., 1980, PROGRAM IMSLEXY: INVERSION OF EX AND EY
C          FREQUENCY SOUNDINGS FROM A GROUNDED WIRE SOURCE, USGS OPEN-
C          FILE REPT. 80-1073.
C
C***********************************************************************
C
C****  THE USER MUST DECLARE THE CALLING PARAMETERS AS EXTERNAL IN THE
C      CALLING PROGRAM (ANY DESIRED NAMES MAY BE USED).
C E.G.,
C
C [MAIN]:
C     EXTERNAL MY_FCODE,MY_PCODE,MY_SUBZ,MY_SUBEND
C     CALL NLSOL(MY_FCODE,MY_PCODE,MY_SUBZ,MY_SUBEND)
C     STOP   !<OR USE>: CALL EXIT
C     END
C [FCODE]:
C     SUBROUTINE MY_FCODE(Y,X,B,W,F,IN,IDER)
C     USER WRITTEN TO EVALUATE THE NONLINEAR OBJECTIVE FUNCTION (F)
C     USED IN NLSOL AS THE WEIGHTED SUM OF (Y(IN)-F)**2, WHERE
C     Y= OBSERVED DEPENDENT VARIABLE ARRAY (DIM. N, WHERE N IS
C      GIVEN IN $PARMS NAMELIST INPUT--SEE BELOW).
C     X= OBSERVED INDEPENDENT VARIABLE ARRAY (DIM. N,M, WHERE
C      M IS IN $PARMS INPUT).
C     B= CURRENT PARAMETER ESTIMATES (DIM. K, WHERE
C      K IS IN $PARMS INPUT).
C     W= WORK ARRAY (DIM. 5)--MAY BE USED TO PASS DATA TO PCODE.
```

```
C      F= (OUTPUT) THE FUNCTION VALUE EVALUATED FOR THE GIVEN
C       Y,X, AND B ARRAYS AT THE OBSERVATION NO. 'IN'.
C      IN= (INPUT) OBSERVATION NO. TO EVALUATE F (1.LE.IN.LE.N),
C       WHICH IS CONTROLLED EXTERNALLY BY 'NLSOL'. USUALLY,
C       IN=1,2,...,N--BUT NOT ALWAYS.
C      IDER= 0 IF ANALYTICAL DERIVATIVES ARE USED (PCODE CALLED
C         AFTER FCODE).
C          = 1 IF ESTIMATED DERIVATIVES ARE USED (PCODE NOT CALLED
C         AFTER FCODE).
C      DIMENSION Y(1),X(500,5),B(1),W(5)
C>>>>> INSERT USER CODE HERE TO EVALUATE F  <<<<<
C      END
C [PCODE]: >> PCODE MAY BE A DUMMY NAME IF ONLY IDER=1 IS TO BE USED. <<
C      SUBROUTINE MY_PCODE(P,X,B,W,F,IN,IP,IB)
C      USER WRITTEN TO EVALUATE THE ANALYTICAL PARTIAL DERIVATIVES OF
C      F WITH RESPECT TO B(J),J=1,2,...,K, AT OBSERVATION 'IN', WHERE
C      P= (OUTPUT) PARTIAL DERIVATIVE ARRAY (DIM. K, WHERE
C       K IS IN $PARMS INPUT).
C      X,B,W ARE THE SAME AS USED IN FCODE (SEE ABOVE).
C      F= LAST FUNCTION VALUE FROM FCODE AT OBSERVATION IN.
C       (NOTE THAT F MAY NOT BE NEEDED, BUT IS AVAILABLE ANYWAY)
C      IN= (INPUT) OBSERVATION NO. TO EVALUATE P ARRAY, WHICH IS
C       CONTROLLED EXTERNALLY BY 'NLSOL' (1.LE.IN.LE.N).
C      IP= (INPUT) THE NO. OF B-PARAMETERS HELD FIXED IN THE LEAST-
C       SQUARES (0.LE.IP.LE.K-1; USE IP=0 IF NONE).
C      IB= ARRAY OF B-PARAMETER INDICES HELD FIXED IF IP.GT.0.
C       NOTE THAT THE INDICES IN IB ARRAY MAY BE IN ANY ORDER,
C       BUT MUST BE BETWEEN 1 AND K (K IS IN $PARMS INPUT).
C      DIMENSION P(1),X(500,5),B(1),W(5),IB(1)
C>>>>>  INSERT USER CODE HERE TO EVALUATE P  <<<<<
C      END
C [SUBZ]:
C      SUBROUTINE MY_SUBZ(Y,X,B,W,NW,N,TITLE,IOUT)
C      USER WRITTEN INITIALIZATION ROUTINE (CALLED ONCE BY 'NLSOL').
C      SUBZ MAY BE USED TO CHECK Y(IN),X(IN,M) AFTER INPUT VIA
C      OBJECT (RUN)-TIME INPUT (SEE BELOW) ON UNIT IALT. ALSO, SUBZ
C      MAY BE USED TO READ ADDITIONAL $INIT PARAMETERS, AND TO LOAD
C       ANY COMMON BLOCKS IF NEEDED IN THE USERS FCODE,PCODE.
C      Y,X,B,W  ARE THE SAME AS USED IN FCODE (SEE ABOVE).
C      NW= USE ANY DUMMY INTEGER VARIABLE (THIS IS
C       TO MAINTAIN COMPATIBILITY WITH 'MARQRT' OR 'IMSLMQ').
C      N= NO. OF OBSERVATIONS IN Y(N),X(N,M) ARRAYS, WHERE
C       K.GE.N.LE.500 (N,M,K  ARE IN $PARMS INPUT).
C      TITLE= (INPUT) 80-CHARACTER HEADING (SEE INPUT FOR005 BELOW).
C      IOUT= 1 IF TO WRITE OUTPUT ON BOTH FOR006 AND FOR016.
C          = 0 IF TO WRITE OUTPUT ONLY ON FOR006.
C      DIMENSION Y(1),X(500,5),B(1),W(5)
C      CHARACTER*80 TITLE
C>>>>> INSERT USER CODE HERE FOR ANY INITIALIZATION DESIRED  <<<<<
C      END
C [SUBEND]:
C      SUBROUTINE MY_SUBEND(Y,X,B,K,N,TITLE,IOUT)
C      USER WRITTEN TERMINATION ROUTINE (CALLED ONCE BY 'NLSOL').
C      SUBEND MAY BE USED TO OUTPUT THE FINAL SOLUTION VECTOR B(I),
C      I=1,2,...,K, IN OTHER FORMS, ETC., AS DESIRED. [OR IT MAY BE A
C      DUMMY ROUTINE; I.E., JUST RETURNS.)
C      Y,X,K,N,TITLE,IOUT  ARE THE SAME AS IN SUBZ AND FCODE.
C      B= (INPUT) IS THE FINAL SOLUTION VECTOR AS DETERMINED BY
C       'NLSOL' (SEE REF1 FOR DETAILS).
C      DIMENSION Y(1),X(500,5),B(1)
C      CHARACTER*80 TITLE
C>>>>>  INSERT USER CODE HERE FOR ANY TERMINATION SUMMARY DESIRED  <<<<<
C      END
C
C***********************************************************************
C
C** INPUT ORDER ON FOR005 (PARAMETER FILE LOGICAL NAME):
C
C 1.   TITLE (MAX. 80-CHARACTERS--ALWAYS READ BEFORE $PARMS INPUT).
C 2.   $PARMS (SAME DEFINITIONS AS IN 'MARQRT', REF2, OR IN 'IMSLMQ',
C      REF3), WHICH INCLUDES: N,K,IP,M,IALT,ISTOP,IWT,IDER,
C      IPRT,NITER,IOUT,SP,B(),IB(); PLUS THE FOLLOWING PARAMETERS FROM
C      REF1 (NL2SOL), P.31-35: IV(),V();  IN ADDITION, THE LOWER AND
C      UPPER BOUND ARRAYS BL(),BH(), RESPECTIVELY, ARE REQUIRED IF
C      SP>2.
C 3.   (OBJECT-RUN-TIME FORMAT STATEMENT) TO DESCRIBE THE FORMAT OF THE
C      DATA MATRIX ROW Y(I),(X(I,J),J=1,M*) READ ON FILE IALT, WHERE
C      M*=M (IF IWT=0) OR M*=M+1 (IF IWT>0), M.LE.4, AND I=1,2,....,N.
C (3A). INSERT DATA MATRIX HERE ONLY IF IALT=5.
C 4.   $INIT OPTIONAL NAMELIST USED FOR READING PROBLEM-DEPENDENT
C      PARAMETERS USED IN SUBROUTINE SUBZ (SEE ABOVE).  CURRENTLY,
C      THE FOLLOWING $INIT NAMES (AND DIM.) CAN BE USED: IOB,MM,X0,Y0,
C      L,EP,EPS,NEPS,METHOD,NFIN,IER,MEV,IOPT,NSIG,MAXFN,DELTA,PARM(4),
C      AND IRATIO(2).
C 5.   OPTIONALLY, REPEAT STEPS 1-4, IF PARAMETER ISTOP=0 WAS USED
C      IN THE LAST STEP 2.
C
C** OUTPUT IS GIVEN ON FOR006 (ON-LINE USUALLY) AND ON FOR016(IF IOUT=1)
C    FOR016 CONTAINS ALL PRINTABLE OUTPUT SELECTED VIA $PARMS IPRT,IOUT.
C    NOTE: IPRT=0 GIVES ABBREVIATED OUTPUT ON FOR006 (BUT MORE ON FOR016)
C          IPRT=1 OR -2 GIVES DETAILED OUTPUT ON BOTH 6 AND 16.
C          IPRT=-1 GIVES MODERATE OUTPUT ON 6 (DETAILED ON 16).
C
C** TO RUN ON VAX (ELIMINATE <> DELIMITERS IN SUBSTITUTIONS):
C
C    $ASSIGN <PARAMETER FILE NAME> FOR005
C    $ASSIGN <DATA MATRIX FILE NAME> FOR010
C    $RUN <MAIN NAME>
C
C***********************************************************************
C
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$ CHANGE THE FOLLOWING FORTRAN-77 PARAMETER STATEMENT ONLY IF
C$$ INCREASING THE DEFAULT DIMENSIONS FOR NLSOL:
      PARAMETER (NDIM=500,MDIM=5,KDIM=20)
C$$ WHERE NDIM=MAX.OBS., MDIM=MAX.INDEP.VARS., KDIM=MAX.UNKNOWN PARMS.
C$$ DO NOT CHANGE THE FOLLOWING RELATED PARAMETER STATEMENT:
      PARAMETER (K1DIM=KDIM-1,K2DIM=KDIM+KDIM,M1DIM=MDIM-1,
     1 IVDIM=KDIM+60,NKVDIM=96+2*NDIM+(KDIM*(7*KDIM+41))/2)
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
C
      REAL*4 L
      DIMENSION B(KDIM),SQWT(NDIM),IB(K1DIM),C(KDIM),INDEX(KDIM),
     1 IV(IVDIM),V(NKVDIM),CBOUND(K2DIM),
     2 BL(KDIM),BH(KDIM),CL(KDIM),CH(KDIM),SE(KDIM),
     3 W(KDIM),PARM(4),IRATIO(2),PRNT(5)
      INTEGER SP,SCALEP,SY,SCALEY
      CHARACTER*3 CHAR3
      CHARACTER*6 CALLED
      CHARACTER*80 TITLE
      CHARACTER*132 LINE132
      CHARACTER*72 FMT
      COMMON/FIXDAT/Y(NDIM),X(NDIM,MDIM),BFIX(KDIM),IIB(K1DIM),IIP,
     1 IDER_,K_,ISP
      COMMON/BOUNDS/BL_(KDIM),BH_(KDIM)
      COMMON/REVCOM/R(NDIM)
      EQUIVALENCE (SQWT(1),X(1,MDIM)),(N,NOBS),(K,KPARMS),(M,MVARS),
     1 (CL(1),CBOUND(1)),(CH(1),CBOUND(KDIM+1))
      EXTERNAL FCODE,PCODE,CALCR
C**
C  THE FOLLOWING COMMON/NAME_LIST/ IS TO SIMULATE ON VAX-11/780:
C      NAMELIST/PARMS/ & READ(5,PARMS) VIA 'CALL NAMELIST(5,'$PARMS',*)'
C      NAMELIST/INIT/ & READ(5,INIT)  VIA 'CALL NAMELIST(5,'$INIT',*)'
C** SEE SUBROUTINE NAMELIST FOR MORE DETAILS, AND ALSO REF1-REF3 FOR
C   DETAILS ON EACH PARAMETER DEFINITION.
C**
      COMMON/NAME_LIST/N,K,IP,M,IALT,ISTOP,IWT,IDER,IPRT,NITER,INON,
     1 FF,T,E,TAU,XL,MODLAM,GAMCR,DEL,ZETA,IOUT,SP,SCALEP,SY,SCALEY,
     2 B,IB, IOB,MM,X0,Y0,L,EP,EPS,NEPS,METHOD,NFIN,IER,MEV,
     3 IV,V,BL,BH,
     4 IOPT,NSIG,MAXFN,DELTA,PARM, H,IRATIO
C**
C  NOTE THAT COMMON/NAME_LIST/ CONTAINS SOME PARAMETERS ONLY FOR
C  COMPATIBILITY WITH 'MARQRT' OR 'IMSLMQ'; I.E., THE FOLLOWING LIST
C  OF PARAMETERS ARE CURRENTLY NOT USED DIRECTLY BY 'NLSOL':
C    INON,FF,T,TAU,XL,MODLAM,GAMCR,DEL,E,ZETA,SY,SCALEY,SCALEP,
C    IOPT,NSIG,MAXFN,DELTA,PARM.
C**
C
C** READ NLSOL TITLE LINE
      READ(5,10,ERR=9000,END=9010) TITLE
10    FORMAT(A80)
C
C**PRESET DEFAULT PARMS (SOME MUST BE GIVEN IN $PARMS ELSE AN ERROR)
C
      N=0
      K=0
      IP=0
      M=0
      IALT=10
      ISTOP=1
      ICALL=1
      IWT=0
      IDER=0
      IPRT=0
      NITER=10
      IOUT=1
      SP=0
      DO 20 I=1,KDIM
      IF(I.LT.KDIM) IB(I)=0
      BL(I)=0.0
      B(I)=0.0
      BH(I)=0.0
20    CONTINUE
22    IV(1)=10
C**
C  PRESET NLITR
C**
      CALL DFAULT(IV,V)
C**
C** OVERRIDE FOR IV(15)=3 DEFAULT (MAY BE CHANGED VIA $PARMS INPUT)
C**
      IV(15)=3
C**
C  READ $PARMS ON FOR005 VIA 'CALL NAMELIST' ON VAX
C**
30    CALL NAMELIST(5,'$PARMS',*9020)
C**
C  SET EQUIVALENT PARAMETERS IN DIFFERENT COMMON'S
C**
      ISP=SP
      DO 32 I=1,KDIM
      BFIX(I)=B(I)
      BL_(I)=BL(I)
      BH_(I)=BH(I)
      IF(I.LT.KDIM) IIB(I)=IB(I)
32    CONTINUE
      IIP=IP
      IDER_=IDER
      K_=K
C**
C  TEST $PARMS BEFORE PROCEEDING
C**
      IF(IP.LT.0.OR.IP.GT.K1DIM)CALL ERRMSG('IP<0 OR IP>19',0,6,16)
      KIP=K-IP
      IF(N.LT.1.OR.N.GT.NDIM.OR.N.LT.KIP)
     1 CALL ERRMSG('N<1,N>500,OR N<K-IP',0,6,16)
      IF(K.LT.1.OR.K.GT.KDIM.OR.KIP.LT.1)
     1 CALL ERRMSG('K<1,K>20,OR K-IP<1',0,6,16)
      IF(M.LT.1.OR.M.GT.M1DIM)CALL ERRMSG('M<1 OR M>4',0,6,16)
      IF(IALT.EQ.6.OR.IALT.EQ.13.OR.IALT.EQ.16.OR.IALT.EQ.4)
     1 CALL ERRMSG('IALT=4,6,13,OR 16',0,6,16)
      IF(ISTOP.EQ.0.AND.IALT.EQ.5)
     1   CALL ERRMSG('ISTOP=0 BUT IALT=5',0,6,16)
      IF(IWT.LT.0.OR.IWT.GT.2)CALL ERRMSG('IWT<0 OR IWT>2',0,6,16)
      IF(IDER.LT.0.OR.IDER.GT.1)CALL ERRMSG('IDER<0 OR IDER>1',0,6,16)
      IF(SP.LT.0.OR.SP.GT.4)CALL ERRMSG('SP<0 OR SP>4',0,6,16)
      IF(IP.GT.0) THEN
        DO J=1,IP
          IF(IB(J).LT.1.OR.IB(J).GT.K) THEN
```

```
            ENCODE(3,43,CHAR3) J
            CALL ERRMSG('IP>0 AND IB(J)<1 OR IB(J)>K FOR J='//
     1      CHAR3,0,6,16)
          ENDIF
        ENDDO
      ENDIF
      IF(SP.EQ.0.OR.SP.EQ.2) GO TO 41
      DO 40 I=1,KPARMS
        IF(SP.EQ.1) THEN
            IF(IP.GT.0) THEN
            DO 42 J=1,IP
              IF(I.EQ.IB(J)) GO TO 40
42          CONTINUE
            ENDIF
          IF(B(I).LE.0.) THEN
            ENCODE(3,43,CHAR3) I
43           FORMAT(I2,'.')
              CALL ERRMSG('SP=1 AND B(I)<=0 FOR I='//CHAR3,0,6,16)
             ENDIF
        ELSE IF(SP.GT.2) THEN
          IF(B(I).LT.BL(I).OR.B(I).GT.BH(I).OR.BL(I).GT.BH(I)) THEN
           ENCODE(3,43,CHAR3) I
            CALL ERRMSG('SP>2 AND B(I)<BL(I), '//
     1      'B(I)>BH(I), OR BL(I)>BH(I)'//
     2      ' FOR I='//CHAR3,0,6,16)
           ENDIF
         IF(BL(I).EQ.BH(I)) THEN
            IF(IP.GT.0) THEN
              DO 45 J=1,IP
                IF(I.EQ.IB(J)) GO TO 40
45          CONTINUE
             ENDIF
            ENCODE(3,43,CHAR3) I
            CALL ERRMSG('SP>2 AND BL(I)=BH(I) BUT B(I) NOT HELD '//
     1      'FIXED FOR I='//CHAR3,0,6,16)
           ENDIF
        ENDIF
40    CONTINUE
41    IF(IV(1).EQ.10) THEN
C**
C  NOTE CALL DFAULT(IV,V) WAS PRESET BEFORE SPARMS READ
C**
        IV(18)=NITER
        IF(IPRT.GT.-3.AND.IPRT.LT.1) THEN
          IV(19)=-1
        ELSE
          IV(19)=IPRT
        ENDIF
        IF(IOUT.EQ.0) THEN
          IV(21)=6
        ELSE
          IV(21)=16
        ENDIF
      ENDIF
      IF(IP.GT.0) THEN
```

```
        DO 50 I=1,IP
          IF(IB(I).LE.0)CALL ERRMSG('IP>0 BUT SOME IB(I)<=0',0,6,16)
50      CONTINUE
      ENDIF
C
C  READ OBJECT(RUN)-TIME FORMAT FOR DATA MATRIX FROM FILE IALT.
C
      READ(5,60,ERR=9000,END=9010) FMT
60    FORMAT(A72)
      IF(IWT.EQ.0) THEN
        M1=MVARS
      ELSE
        M1=MVARS+1
      ENDIF
      DO 70 I=1,NOBS
        READ(IALT,FMT,ERR=9030,END=9040) Y(I),(X(I,J),J=1,M1)
        IF(IWT.EQ.0.OR.X(I,M1).EQ.0.0) THEN
          SQWT(I)=1.0
          GO TO 70
        ELSE IF(IWT.EQ.1) THEN
          SQWT(I)=1.0/X(I,M1)
        ELSE
          SQWT(I)=1.0/SQRT(ABS(X(I,M1)))
        ENDIF
70    CONTINUE
C
C  INITIALIZE VIA CALL SUBZ (READ $INIT AND TEST, LOAD COMMON, ETC.)
C
      CALL SUBZ(Y,X,BFIX,PRNT,NPRNT,N,TITLE,IOUT)
C     ****************************************
C
C  WRITE SPARMS ON FOR006 AND FOR016 (THE LATTER IF IOUT=1)
C
      CALL NONBLANK(TITLE,NB)
      WRITE(6,80) TITLE,N,K,IP,M,IALT,ISTOP,IWT,IDER,IPRT,NITER,IOUT,SP
80    FORMAT('1(NLSOL):',8X,A<NB>//' N=',4X,I6,T18,'K=',4X,I6,T34,'IP=',
     1 3X,I6,T50,'M=',4X,I6,T66,'IALT=',1X,I6/' ISTOP=',I6,T18,'IWT=',
     2 2X,I6,T34,'IDER=',I7,T50,'IPRT=',I7,T66,'NITER=',I6/' IOUT=',
     3 5X,I2,T18,'SP=',3X,I6)
      IF(IOUT.NE.0)
     1WRITE(16,80)TITLE,N,K,IP,M,IALT,ISTOP,IWT,IDER,IPRT,NITER,IOUT,SP
      IF(IP.GT.0) THEN
        WRITE(6,90) (IB(I),I=1,IP)
90      FORMAT(/' PARAMETERS HELD FIXED: IB=',20I3)
        IF(IOUT.NE.0) WRITE(16,90) (IB(I),I=1,IP)
      ENDIF
      CALL NONBLANK(FMT,NB)
      WRITE(6,100) FMT
100   FORMAT(/' FMT=',A<NB>/)
      IF(IOUT.NE.0) WRITE(16,100) FMT
      IF(SP.GT.2) THEN
        WRITE(6,111) (BL(I),I=1,KPARMS)
111     FORMAT(/' PARAMETER LOWER BOUNDS: BL='//(5E16.8))
        IF(IOUT.NE.0) WRITE(16,111) (BL(I),I=1,KPARMS)
      ENDIF
      WRITE(6,110) (B(I),I=1,KPARMS)
110   FORMAT(/' INITIAL PARAMETERS: B='//(5E16.8))
      IF(IOUT.NE.0) WRITE(16,110) (B(I),I=1,KPARMS)
      IF(SP.GT.2) THEN
        WRITE(6,112) (BH(I),I=1,KPARMS)
112     FORMAT(/' PARAMETER HIGHER BOUNDS: BH='//(5E16.8))
        IF(IOUT.NE.0) WRITE(16,112) (BH(I),I=1,KPARMS)
      ENDIF
      DO 120 I=1,KDIM
120     INDEX(I)=I
      IF(IP.EQ.0) THEN
        DO 130 I=1,KPARMS
        IF(SP.GT.2) THEN
          CL(I)=BL(I)
          CH(I)=BH(I)
        ENDIF
130     C(I)=B(I)
      ELSE
C
C  REORDER B TO C WHEN IP>0 (AND BL,BH TO CL,CH, RESPECTIVELY)
C
        IM=0
        DO 150 I=1,KPARMS
        DO 140 J=1,IP
          IF(I.EQ.IB(J)) GO TO 150
140     CONTINUE
        IM=IM+1
        C(IM)=B(I)
        IF(SP.GT.2) THEN
          CL(IM)=BL(I)
          CH(IM)=BH(I)
        ENDIF
        INDEX(IM)=I
150     CONTINUE
        WRITE(6,160) (I,I=1,KPARMS)
160     FORMAT(/' PARAMETER INDEX:',20I3)
        IF(IOUT.NE.0) WRITE(16,160) (I,I=1,KPARMS)
        WRITE(6,170) (INDEX(I),I=1,KIP)
170     FORMAT(' REORDERED AS...:',20I3)
        IF(IOUT.NE.0) WRITE(16,170) (INDEX(I),I=1,KIP)
        WRITE(6,180) (C(I),I=1,KIP)
180     FORMAT(/' REORDERED PARAMETERS:'//(5E16.8))
        IF(IOUT.NE.0) WRITE(16,180) (C(I),I=1,KIP)
      ENDIF
C
C  PERFORM INITIAL PARAMETER TRANSFORMS VIA SP (SCALEP)
C
      IF(SP.EQ.0) GO TO 220
      DO 210 I=1,KIP
        GO TO (201,202,203,203),SP
201     C(I)=ALOG(C(I))
        GO TO 210
202     C(I)=ASINH(C(I))
        GO TO 210
203     TEM=(C(I)-CL(I))/(CH(I)-CL(I))
```

```
        IF(SP.EQ.3) THEN
          C(I)=ASIN(SQRT(TEM))
        ELSE
          C(I)=ERFINV(2.0*TEM-1.0)
        ENDIF
210   CONTINUE
C
C  INTERFACE WITH NL2ITR USING MARQRT FCODE AND PCODE (IF IDER=0)
C
220   ENCODE(6,222,CALLED) ICALL
222   FORMAT(I3,' **')
      WRITE(6,221) CALLED
221   FORMAT('0** NLITR (IDER=0) OR NL2SNO (IDER=1) CALLED:',A6/)
      IF(IOUT.NE.0) WRITE(16,221) CALLED
      IF(IDER.EQ.0) THEN
        CALL NLITR(NOBS,KIP,C,IV,V,CBOUND,FCODE,PCODE)
C       ********************************************
      ELSE
        CALL NL2SNO(NOBS,KIP,C,CALCR,IV,V,IDUMMY,CBOUND,FCODE)
C       ****************************************************
      ENDIF
C
C  GET INVERSE PARAMETER TRANSFORMATION OF SOLUTION VECTOR C
C
      IF(SP.EQ.0) GO TO 229
      DO 228 I=1,KIP
        GO TO (224,225,226,226),SP
224     C(I)=EXP(C(I))
        GO TO 228
225     C(I)=SINH(C(I))
        GO TO 228
226     TEM=CH(I)-CL(I)
        IF(SP.EQ.3) THEN
          C(I)=CL(I)+TEM*SIN(C(I))**2
        ELSE
          C(I)=CL(I)+0.5*TEM*(1.0+ERF(C(I)))
        ENDIF
228   CONTINUE
C
C  OUTPUT SELECTED RESULTS ON FOR006 (ALL RESULTS ON FOR016 IF IOUT=1)
C
229   IF(IOUT.NE.0.AND.IPRT.NE.0) THEN
        I=1
        REWIND 16
230     READ(16,232,END=240) LINE132
232     FORMAT(A)
        IF(I.EQ.1) THEN
C
C  VAX FUNCTION 'LIB$INDEX' USED TO DISTINGUISH FROM ARRAY 'INDEX'
C
          IF(LIB$INDEX(LINE132,'CALLED:'//CALLED).EQ.0) GO TO 230
          I=0
          GO TO 230
        ENDIF
        IF(LIB$INDEX(LINE132,'OBS.Y(I)').NE.0) GO TO 236
```

```
          IF(LIB$INDEX(LINE132,'COVARIANCE = SCALE').NE.0) GO TO 236
          CALL NONBLANK(LINE132,J)
          IF(J.LE.0) GO TO 230
          WRITE(6,234) LINE132
234       FORMAT(A<J>)
          GO TO 230
236       READ(16,232,END=240) LINE132
          GO TO 236
        ENDIF
240     IF(IOUT.NE.0) WRITE(16,250)
250     FORMAT(/3X,'I',4X,'OBS.Y(I)',6X,'CAL',11X,'RES',8X,
     1 '%RES.ERR',6X,'X(I,1)',8X,
     2 'X(I,2)',8X,'X(I,3)',8X,'X(I,4)',8X,'WT(I)')
        IF(IPRT.EQ.-2) WRITE(6,250)
        SUMF2=0.0
        IF(IDER.NE.0) IADR=IV(50)-1
        DO 270 I=1,NOBS
          IF(IDER.EQ.0) THEN
            F2=R(I)
          ELSE
            F2=V(IADR+I)
          ENDIF
          RES=F2/SQWT(I)
          CAL=Y(I)-RES
          IF(CAL.NE.0.0) THEN
            PERR=100.0*RES/ABS(CAL)
          ELSE
            PERR=0.0
          ENDIF
          WT=SQWT(I)**2
          SUMF2=SUMF2+RES**2
          IF(IPRT.EQ.-2)WRITE(6,260) I,Y(I),CAL,RES,PERR,
     1 (X(I,J),J=1,4),WT
260       FORMAT(1X,I3,2E14.6,E11.3,6E14.6)
          IF(IOUT.NE.0) WRITE(16,260) I,Y(I),CAL,RES,PERR,
     1 (X(I,J),J=1,4),WT
270     CONTINUE
        IF(NOBS.EQ.KIP) THEN
          RMSERR=0.0
        ELSE
          RMSERR=SQRT(SUMF2/(NOBS-KIP))
        ENDIF
        WRITE(6,280) RMSERR
280     FORMAT(/' ** RMSERR=',E16.8)
        IF(IOUT.NE.0) WRITE(16,280) RMSERR
        IF(IV(26).LE.0) GO TO 380
C
C  A COVARIANCE MATRIX WAS COMPUTED (GET ADDITIONAL STATISTICS)
C
        IADR=IV(26)-1
        IF(IPRT.LT.-1) WRITE(6,290)
290     FORMAT(/' COVARIANCE MATRIX')
        DO 320 I=1,KIP
        DO 300 J=1,I
300       W(J)=V(IADR+LOC(J,I))
```

```
        SE(I)=SQRT(ABS(W(I)))
        IF(IPRT.LT.-1) WRITE(6,310) INDEX(I),(W(J),J=1,I)
310     FORMAT(1X,I2,10E12.4/(3X,10E12.4))
320   CONTINUE
C
C  GET CORRELATION COEFFICIENT MATRIX
C
      IF(IOUT.NE.0) WRITE(16,330)
330   FORMAT(/' CORRELATION MATRIX')
      IF(IPRT.LT.0) WRITE(6,330)
      DO 350 I=1,KIP
        IF(SE(I).EQ.0.0) THEN
          W(I)=1.0
        ENDIF
      DO 340 J=1,I
        IF(SE(J).NE.0.0) W(J)=V(IADR+LOC(J,I))/(SE(I)*SE(J))
340   CONTINUE
      IF(IOUT.NE.0) WRITE(16,310) INDEX(I),(W(J),J=1,I)
      IF(IPRT.LT.0) WRITE(6,310) INDEX(I),(W(J),J=1,I)
350   CONTINUE
C
C  PRINT PARAMETER STANDARD ERRORS (SE) AND RELATIVE ERRORS
C
      WRITE(6,360)
360   FORMAT(/'    **PARM_SOL.   STD_ERROR   REL_ERROR    % ERROR **'/)
      IF(IOUT.NE.0) WRITE(16,360)
      DO 370 I=1,KIP
        RELERR=0.0
        IF(C(I).NE.0.0) RELERR=SE(I)/C(I)
        PERR=100.*RELERR
        WRITE(6,310) INDEX(I),C(I),SE(I),RELERR,PERR
        IF(IOUT.NE.0) WRITE(16,310) INDEX(I),C(I),SE(I),RELERR,PERR
370   CONTINUE
C
C  PUT SOLUTION C AND BFIX TOGETHER (IF IP>0)
C
380   DO 390 I=1,KIP
390   W(I)=C(I)
      IF(IP.EQ.0) GO TO 420
      IM=0
      DO 410 I=1,KPARMS
        W(I)=BFIX(I)
      DO 400 J=1,IP
        IF(I.EQ.IB(J)) GO TO 410
400   CONTINUE
        IM=IM+1
        W(I)=C(IM)
410   CONTINUE
420   CALL SUBEND(Y,X,W,K,N,TITLE,IOUT)
C       ********************************
      IF(ISTOP.NE.1) THEN
        READ(5,10,ERR=9000,END=9010) TITLE
        IF(IALT.NE.5) REWIND IALT
        ICALL=ICALL+1
        GO TO 22
      ENDIF
C
C** RETURN FROM NLSOL
C
      RETURN
9000  CALL ERRMSG('ERR=9000 READING FOR005',0,6,16)
9010  CALL ERRMSG('PREMATURE E.O.F (END=9010) READING FOR005',0,6,16)
9020  CALL ERRMSG('END *9020 READING FOR005 IN (NAMELIST)',0,6,16)
9030  CALL ERRMSG('END=9030 READING FILE IALT',0,6,16)
9040  CALL ERRMSG('PREMATURE E.O.F (END=9040) READING FILE IALT',
     1 0,6,16)
C
C** END OF SUBROUTINE NLSOL
C
      END
      SUBROUTINE NLITR(N,KIP,C,IV,V,CBOUND,FCODE,PCODE)
C
C**CALCULATES BOTH THE RESIDUAL VECTOR R(N) & ANALYTICAL JACOBIAN
C  JAC(N,KIP) BY 'REVERSE COMMUNICATION VIA INTERNAL CALL NL2ITR'
C  (SEE REF1, P. 38).
C
C     N = NO. OBSERVATIONS <=500 (SEE NDIM BELOW)
C     KIP = NO. ADJUSTABLE PARAMETERS =K-IIP WHERE
C        K=TOTAL PARAMETERS, IIP=NO. PARAMETERS HELD FIXED
C          IN IIB(IIP) VIA COMMON/FIXDAT/
C     C() = I/O PARAMETER VECTOR (SUPPLIED BY NL2ITR)
C        WHICH ARE THE UNCONSTRAINED PARAMETERS IN NL2ITR.
C     IV() = SAME CONTROL INFORMATION SET BY NLSOL (OR NL2ITR).
C     V() = SAME CONTROL INFORMATION SET BY NLSOL (OR NL2ITR).
C     CBOUND = INPUT ARRAY OF LOW AND HIGH BOUNDS USED ONLY WHEN SP>2.
C     FCODE = EXTERNAL FUNCTION NAME (SAME AS USED IN 'MARQRT' OR
C        'IMSLMQ' TO COMPUTE THE NONLINEAR OBJECTIVE FUNCTION).
C     PCODE = EXTERNAL ANALYTIC DERIVATIVE NAME (SAME AS USED IN
C        'MARQRT' WHEN IDER=0) CORRESPONDING TO EACH FCODE CALL.
C
C**SEE REF1 (P.38) FOR MORE DETAILS ON CALLING NL2ITR.
C
C**OTHER DATA IN COMMON/FIXDAT/ MUST BE PRESET.
C
C
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$ CHANGE THE FOLLOWING FORTRAN-77 PARAMETER STATEMENT ONLY IF
C$$ INCREASING THE DEFAULT DIMENSIONS FOR NLSOL:
      PARAMETER (NDIM=500,MDIM=5,KDIM=20)
C$$ WHERE NDIM=MAX.OBS., MDIM=MAX.INDEP.VARS., KDIM=MAX.UNKNOWN PARMS.
C$$ DO NOT CHANGE THE FOLLOWING RELATED PARAMETER STATEMENT:
      PARAMETER (K1DIM=KDIM-1)
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C
      INTEGER SP
      DIMENSION C(1),IV(1),V(1),CBOUND(1),PRNT(5),SQWT(NDIM),
     1 BIP(KDIM),D(KDIM),R(NDIM),PART(KDIM),W(KDIM)
      REAL*4 JAC(NDIM,KDIM)
      COMMON/FIXDAT/Y(NDIM),X(NDIM,MDIM),BFIX(KDIM),IIB(K1DIM),IIP,
     1 IDER,KPARMS,SP
```

```
      COMMON/BOUNDS/BL(KDIM),BH(KDIM)
      COMMON/REVCOM/R
      EQUIVALENCE (SQWT(1),X(1,MDIM))
      DATA NN/NDIM/
C
C  GET INVERSE PARAMETER TRANSFORMATION (C TO BIP)
C
10    CALL INTRAN(KIP,C,CBOUND,BIP)
C
C  DETERMINE FROM IV(1) HOW TO CALL NL2ITR
      IV1=IV(1)
        DO 120 I=1,N
          CALL FCODE(Y,X,BIP,PRNT,F,I,IDER)
C         ********************************
          IF(IV1.NE.2) R(I)=SQWT(I)*(Y(I)-F)
          IF(IV1.EQ.1) GO TO 120
          CALL PCODE(PART,X,BIP,PRNT,F,I,IIP,IIB)
C         ***************************************
C
C  SCALE PART(J) VIA SP AND THE DERIVATIVE CHAIN-RULE.
C
         IF(SP.EQ.0) GO TO 80
          IF(SP.EQ.1) THEN
            DO 11 K=1,KPARMS
11          PART(K)=BIP(K)*PART(K)
          ELSE IF(SP.EQ.2) THEN
            DO 12 K=1,KPARMS
            IF(PART(K).EQ.0.0) GO TO 12
            TEM=BIP(K)+SQRT(BIP(K)**2+1.0)
            PART(K)=0.5*(TEM+1.0/TEM)*PART(K)
12          CONTINUE
          ELSE IF (SP.EQ.3) THEN
            DO 13 K=1,KPARMS
            IF(PART(K).EQ.0.0) GO TO 13
            PART(K)=2.*PART(K)*SQRT((BIP(K)-BL(K))*
     1         (BH(K)-BIP(K)))
13          CONTINUE
          ELSE IF(SP.EQ.4) THEN
            DO 14 K=1,KPARMS
            IF(PART(K).EQ.0.0) GO TO 14
            TEM=BH(K)-BL(K)
            PART(K)=0.56418958*PART(K)*TEM*EXP(-(ERFINV(2.*(BIP(K)-
     1       BL(K))/TEM-1.))**2)
14          CONTINUE
          ENDIF
80        IF(IIP.EQ.0) THEN
            DO 90 J=1,KIP
90          JAC(I,J)=-SQWT(I)*PART(J)
          ELSE
            IM=0
            DO 110 K=1,KPARMS
            DO 100 J=1,IIP
              IF(K.EQ.IIB(J)) GO TO 110
100         CONTINUE
            IM=IM+1
            JAC(I,IM)=-SQWT(I)*PART(K)
110         CONTINUE
          ENDIF
120     CONTINUE
C
C
      CALL NL2ITR(D,IV,JAC,N,NN,KIP,R,V,C)
C     ************************************
      IF(IV(1).EQ.1.OR.IV(1).EQ.2) GO TO 10
      RETURN
      END
      SUBROUTINE INTRAN(KIP,C,CBOUND,BIP)
C
C**INVERSE PARAMETER TRANSFORMATION USED IN 'NLSOL','NLITR'.
C
C  CALCULATES CONSTRAINED PARAMETERS FOR FCODE OR PCODE BACK FROM THE
C  UNCONSTRAINED PARAMETERS IN 'NL2ITR' OR 'NL2SNO'
C
C     KIP = NO. ADJUSTABLE PARAMETERS = K-IIP (IIP IN COMMON/FIXDAT)
C     C() = INPUT UNCONSTRAINED VECTOR (DIM. KIP)
C     CBOUND = INPUT CONSTRAINED BOUNDS, IF ANY.
C     BIP() = OUTPUT CONSTRAINED VECTOR (DIM. KPARMS--IN COMMON).
C
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$ CHANGE THE FOLLOWING FORTRAN-77 PARAMETER STATEMENT ONLY IF
C$$ INCREASING THE DEFAULT DIMENSIONS FOR NLSOL:
      PARAMETER (NDIM=500,MDIM=5,KDIM=20)
C$$ WHERE NDIM=MAX.OBS., MDIM=MAX.INDEP.VARS., KDIM=MAX.UNKNOWN PARMS.
C$$ DO NOT CHANGE THE FOLLOWING RELATED PARAMETER STATEMENT:
      PARAMETER (K1DIM=KDIM-1)
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C
      INTEGER SP
      DIMENSION C(1),CBOUND(1),BIP(1),CTEM(KDIM)
      COMMON/FIXDAT/Y(NDIM),X(NDIM,MDIM),BFIX(KDIM),IIB(K1DIM),IIP,
     1 IDER,KPARMS,SP
      IF(SP.EQ.0) THEN
        DO 10 I=1,KIP
10      CTEM(I)=C(I)
      ELSE
        DO 50 I=1,KIP
          GO TO (20,30,40,40),SP
20        CTEM(I)=EXP(C(I))
          GO TO 50
30        CTEM(I)=SINH(C(I))
          GO TO 50
40        DIF=CBOUND(KDIM+I)-CBOUND(I)
          IF(SP.EQ.3) THEN
            CTEM(I)=CBOUND(I)+DIF*SIN(C(I))**2
          ELSE
            CTEM(I)=CBOUND(I)+0.5*DIF*(1.0+ERF(C(I)))
          ENDIF
50      CONTINUE
      ENDIF
      IF(IIP.EQ.0) THEN
```

```
        DO 60 I=1,KIP
60        BIP(I)=CTEM(I)
      ELSE
        IM=0
        DO 80 I=1,KPARMS
          BIP(I)=BFIX(I)
        DO 70 J=1,IIP
          IF(I.EQ.IIB(J)) GO TO 80
70      CONTINUE
        IM=IM+1
        BIP(I)=CTEM(IM)
80      CONTINUE
      ENDIF
      RETURN
      END
      SUBROUTINE CALCR(N,KIP,C,NF,R,LASTNF,CBOUND,FCODE)
C
C**CALCULATES RESIDUAL VECTOR R(N) FOR 'NL2SNO' WHEN IDER=1.
C
C     N = NO. OBSERVATIONS <=500 (SEE NDIM BELOW)
C     KIP = NO. ADJUSTABLE PARAMETERS =K-IIP WHERE
C        K=TOTAL PARAMETERS, IIP=NO. PARAMETERS HELD FIXED
C          IN IIB(IIP) VIA COMMON/FIXDAT/
C     C() = INPUT PARAMETER VECTOR (SUPPLIED BY NL2SNO)
C        WHICH ARE THE UNCONSTRAINED PARAMETERS IN NL2SNO.
C     NF = INVOCATION COUNT (INPUT)FOR USE BY NL2SNO OR NL2SOL.
C     R() = OUTPUT WEIGHTED RESIDUAL VECTOR (DIM. N)
C     LASTNF = LAST NF (ON EXIT FOR POSSIBLE USE IN CALCJ OR NL2SOL).
C     CBOUND = INPUT ARRAY OF LOW AND HIGH BOUNDS USED ONLY WHEN SP>2.
C     FCODE = EXTERNAL FUNCTION NAME (SAME AS USED IN 'MARQRT' OR
C        'IMSLMQ' TO COMPUTE THE NONLINEAR OBJECTIVE FUNCTION).
C
C**OTHER DATA IN COMMON/FIXDAT/ MUST BE PRESET.
C
C
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C$$ CHANGE THE FOLLOWING FORTRAN-77 PARAMETER STATEMENT ONLY IF
C$$ INCREASING THE DEFAULT DIMENSIONS FOR NLSOL:
      PARAMETER (NDIM=500,MDIM=5,KDIM=20)
C$$ WHERE NDIM=MAX.OBS., MDIM=MAX.INDEP.VARS., KDIM=MAX.UNKNOWN PARMS.
C$$ DO NOT CHANGE THE FOLLOWING RELATED PARAMETER STATEMENT:
      PARAMETER (K1DIM=KDIM-1)
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C
      INTEGER SP
      DIMENSION C(1),R(1),CBOUND(1),PRNT(5),SQWT(NDIM),BIP(KDIM)
      COMMON/FIXDAT/Y(NDIM),X(NDIM,MDIM),BFIX(KDIM),IIB(K1DIM),IIP,
     1 IDER,KPARMS,SP
      EQUIVALENCE (SQWT(1),X(1,MDIM))
C
C  GET INVERSE PARAMETER TRANSFORMATION (C TO BIP)
C
      CALL INTRAN(KIP,C,CBOUND,BIP)
C
C  COMPUTE RESIDUAL VECTOR R(N) USING BIP IN FCODE
```

```
C
      DO 10 I=1,N
        CALL FCODE(Y,X,BIP,PRNT,F,I,IDER)
C       ********************************
        R(I)=SQWT(I)*(Y(I)-F)
10    CONTINUE
      LASTNF=NF
      RETURN
      END
      SUBROUTINE NONBLANK(C,NB)
C--DETERMINE NON-BLANK CHAR LENGTH (=NB ON EXIT) OF C*(*)
C  NOTE THAT NB WILL BE IN [0,LEN(C)].
C
      CHARACTER*(*) C
      L=LEN(C)
      DO 10 I=L,1,-1
        NB=I
        IF(C(I:I).NE.' ') RETURN
10    CONTINUE
      NB=0
      RETURN
      END
      SUBROUTINE PROCINFO(ABS_VALUES,INCR_VALUES)
C
C** SUBROUTINE TO OBTAIN ABSOLUTE AND INCREMENTAL VALUES OF PROCESS
C   PARAMETERS: CPU TIME, BUFFERED I/O COUNT, DIRECT I/O COUNT, AND
C   PAGE FAULTS.
C
      IMPLICIT INTEGER*2(W),INTEGER*4(L)
      PARAMETER (JPI$_CPUTIM = '00000407'X,
     1 JPI$_BUFIO = '0000040C'X,JPI$_DIRIO = '0000040B'X,
     2 JPI$_PAGEFLTS= '0000040A'X)
      INTEGER*4 ABS_VALUES(4),INCR_VALUES(4),LCL_VALUES(4)
      COMMON/ITEMLIST/
     1 W_LEN1,W_CODE1,L_ADDR1,L_LENADDR1,
     2 W_LEN2,W_CODE2,L_ADDR2,L_LENADDR2,
     3 W_LEN3,W_CODE3,L_ADDR3,L_LENADDR3,
     4 W_LEN4,W_CODE4,L_ADDR4,L_LENADDR4,
     5 W_LEN5,W_CODE5
      DATA W_LEN1,W_LEN2,W_LEN3,W_LEN4,W_LEN5/5*4/
      DATA W_CODE1/JPI$_CPUTIM/,
     1 W_CODE2/JPI$_BUFIO/,
     2 W_CODE3/JPI$_DIRIO/,
     3 W_CODE4/JPI$_PAGEFLTS/,
     4 W_CODE5/0/
      DATA L_LENADDR1,L_LENADDR2,L_LENADDR3,L_LENADDR4/4*0/
      L_ADDR1=%LOC(LCL_VALUES(1))
      L_ADDR2=%LOC(LCL_VALUES(2))
      L_ADDR3=%LOC(LCL_VALUES(3))
      L_ADDR4=%LOC(LCL_VALUES(4))
C** PERFORM THE SYSTEM SERVICE CALL
      CALL SYS$GETJPI(,,,W_LEN1,,,)
C** ASSIGN THE NEW VALUES TO THE ARGUMENTS
      DO I=1,4
        INCR_VALUES(I)=LCL_VALUES(I)-ABS_VALUES(I)
```

```
          ABS_VALUES(I)=LCL_VALUES(I)
        END DO
        RETURN
        END
        REAL FUNCTION RFLAGS(N,FUN,TOL,TO,TM,T,NEW)
C--FOURIER TRANSFORM LAG CONVOLUTION & SPLINE INTERPOLATION
C  GIVES FOURIER COSINE OR SINE TRANSFORMS VIA RLAGF0,RLAGF1
C  REF: ANDERSON,1975,NTIS REPT. PB-242-800,P.76-87.
C
C        N = 0 FOR COSINE TRANSFORM (VIA RLAGF0)
C        N = 1 FOR SINE TRANSFORM (VIA RLAGF1)
C      FUN = EXTERNAL REAL KERNEL FUNCTION.
C      TOL = TOLERANCE REQUESTED FOR RLAGF0 OR RLAGF1
C       TO = TMIN TO USE (E.G., LET TO=.5*TMIN, TMIN=TRUE)
C       TM = TMAX TO USE (TM>TO)
C        T = TRANSFORM PARAMETER (TO<=T<=TM) FOR THIS CALL (NEW=1 OR 0)
C      NEW = 1 REQUIRED FOR 1ST CALL OR TO RESET SPLINE COEFFICIENTS.
C      NEW = 0 FOR ALL CALLS AFTER 1ST--USES SPLINE INTERPOLATION ONLY.
C
        REAL ARG(200),Y(200),AR(200),BR(200),CR(200),
       & D(2),W1(200),W2(200)
        EXTERNAL FUN
        DATA D/2*0.0/
        IF(NEW.EQ.0) GO TO 3
        NT=AINT(5.*ALOG(TM/TO))+5
        IF(NT.GT.200)CALL ERRMSG('IN RFLAGS: NT>200   ',4,6,16)
        NT1=NT+1
        X0=ALOG(TO)+.2*NT
        NU=1
        DO 1 J=1,NT
        I=NT1-J
        X=X0-.2*J
        EX=EXP(X)
        ARG(I)=EX
        IF(N.EQ.0) Y(I)=RLAGF0(X,FUN,TOL,L,NU)/EX
        IF(N.NE.0) Y(I)=RLAGF1(X,FUN,TOL,L,NU)/EX
1       NU=0
        CALL SPLIN1(NT,0.0,ARG,Y,AR,BR,CR,0,D,W1,W2)
2       IF(NT.LT.0) CALL ERRMSG('IN RFLAGS: NT<0 AFTER SPLIN1  ',6,6,16)
3       IF(T.LT.TO) CALL ERRMSG('IN RFLAGS: T<TO',3,6,16)
        IF(T.GT.TM) CALL ERRMSG('IN RFLAGS: T>TM',3,6,16)
        CALL SPOINT(NT,ARG,Y,AR,BR,CR,T,X)
        RFLAGS=X
        RETURN
        END
        SUBROUTINE SPLIN1(M,H,X,Y,A,B,C,IT,D,P,S)
C--ONE DIMENSIONAL CUBIC SPLINE COEFFICIENT DETERMINATION.
C
C           BY  W.L.ANDERSON, U.S. GEOLOGICAL SURVEY, DENVER, COLORADO
C
C  PARMS--- M= NUMBER OF DATA POINTS .GT. 2
C           H= EQUAL INTERVAL OPTION WHEN H.GT.0. (USE DUMMY X HERE),
C              UNEQUAL INTERVALS IF H=0. (X REQUIRED STORAGE)
C           X= INDEP.VAR WHEN H=0. (DIM .GE. M).
C           Y= DEPENDENT VARIABLE  (DIM .GE. M).
```

```
C          A,B,C=COEFF.ARRAYS (EACH DIM .GE. M)
C                RESULTS ARE RETURNED IN 1ST(M-1) ELEMENTS OF A,B,&C.
C                ALSO USED AS WORK ARRAYS DURING EXECUTION.
C          IT= TYPE OF BOUNDARY CONDITION SUPPLIED IN D ARRAY. USE
C             IT=1 IF 1ST DERIVATIVES GIVEN AT END POINTS, OR
C             IT=0 IF 2ND DERIVATIVES GIVEN AT END POINTS.
C          D= BOUNDARY ARRAY (DIM 2) AT POINT 1 AND M RESPECTIVELY.
C          P,S= WORK ARRAYS (EACH DIM=M).
C--ERROR RETURN WITH M=-(ABS(M)) IF ANY PARM OUT OF RANGE.
C  THE RESULTING CUBIC SPLINE IS OF THE FORM:
C     Y=Y(I)+A(I)*(X-X(I))+B(I)*(X-X(I))**2+C(I)*(X-X(I))**3
C       FOR I=1,2,...,M-1
C
C
      REAL*4  X(1),Y(1),A(1),B(1),C(1),D(2),P(1),S(1),MUL
      IF(IT.LT.0.OR.IT.GT.1.OR.H.LT.0..OR.M.LT.3) GO TO 999
      N=M-1
      IF(IT.EQ.0) GO TO 20
C--1ST DERIVATIVE BOUNDARIES GIVEN
      NE=N-1
      IF(H) 999,11,1
C--EQUAL SPACING H .GT. 0. AND IT=1
    1 HH=3.0/H
      DO 2 I=1,NE
      B(I)=4.0
      C(I)=1.0
      A(I)=1.0
    2 P(I)=HH*(Y(I+2)-Y(I))
      P(1)=P(1)-D(1)
      P(NE)=P(NE)-D(2)
C--SOLUTION OF TRIDIAGONAL MATRIX EQ. OF ORDER NE
    3 C(1)=C(1)/B(1)
      P(1)=P(1)/B(1)
      DO 4 I=2,NE
      MUL=1.0/(B(I)-A(I)*C(I-1))
      C(I)=MUL*C(I)
    4 P(I)=MUL*(P(I)-A(I)*P(I-1))
C--OBTAIN SPLINE COEFFICIENTS
      A(NE+IT)=P(NE)
      I=NE-1
    5 A(I+IT)=P(I)-C(I)*A(I+IT+1)
      I=I-1
       IF(I.GE.1) GO TO 5
      IF(IT.EQ.0) GO TO 6
      A(1)=D(1)
      A(M)=D(2)
    6 IF(H.EQ.0.) GO TO 14
      HH=1.0/H
      DO 7 I=1,N
      MUL=HH*(Y(I+1)-Y(I))
      B(I)=HH*(3.0*MUL-(A(I+1)+2.0*A(I)))
    7 C(I)=HH*HH*(-2.0*MUL+A(I+1)+A(I))
      RETURN
C--UNEQUAL SPACING H=0.. AND IT=1
   11 DO 12 I=1,N
   12 S(I+1)=X(I+1)-X(I)
      DO 13 I=1,NE
      B(I)=2.0*(S(I+1)+S(I+2))
      C(I)=S(I+1)
      A(I)=S(I+2)
   13 P(I)=3.0*(S(I+1)**2*(Y(I+2)-Y(I+1))+S(I+2)**2*(Y(I+1)-Y(I)))/
     $ (S(I+1)*S(I+2))
      P(1)=P(1)-S(3)*D(1)
      P(NE)=P(NE)-S(N)*D(2)
      GO TO 3
   14 DO 15 I=1,N
      HH=1.0/S(I+1)
      MUL=(Y(I+1)-Y(I))*HH**2
      B(I)=3.0*MUL-(A(I+1)+2.0*A(I))*HH
   15 C(I)=-2.0*MUL*HH+(A(I+1)+A(I))*HH**2
      RETURN
C--2ND DERIVATIVE BOUNDARIES GIVEN
   20 NE=N+1
      IF(H) 999,31,21
C--EQUAL SPACING H .GT. 0 AND IT=0
   21 HH=3.0/H
      DO 22 I=2,N
      B(I)=4.0
      C(I)=1.0
      A(I)=1.0
   22 P(I)=HH*(Y(I+1)-Y(I-1))
      B(1)=2.0
      B(NE)=2.0
      C(1)=1.0
      C(NE)=1.0
      A(NE)=1.0
      P(1)=HH*(Y(2)-Y(1))-0.5*H*D(1)
      P(NE)=HH*(Y(M)-Y(N))+0.5*H*D(2)
      GO TO 3
C--UNEQUAL SPACING H=0 AND IT=0
   31 DO 32 I=1,N
   32 S(I+1)=X(I+1)-X(I)
      N1=N-1
      DO 33 I=1,N1
      B(I+1)=2.0*(S(I+1)+S(I+2))
      C(I+1)=S(I+1)
      A(I+1)=S(I+2)
   33 P(I+1)=3.0*(S(I+1)**2*(Y(I+2)-Y(I+1))+S(I+2)**2*(Y(I+1)-Y(I)))/
     *       (S(I+1)*S(I+2))
      B(1)=2.0
      B(NE)=2.0
      C(1)=1.0
      C(NE)=1.0
      A(NE)=1.0
      P(1)=3.0*(Y(2)-Y(1))/S(2)-0.5*S(2)*D(1)
      P(NE)=3.0*(Y(M)-Y(N))/S(M)+0.5*S(M)*D(2)
      GO TO 3
  999 M=-IABS(M)
      RETURN
      END
      SUBROUTINE SPOINT(M,X,Y,A,B,C,XX,YY)
C--GIVEN CUBIC SPLINE COEFF'S A,B,C,AND M OBS.DATA ARRAYS X,Y
C  SPOINT EVALUATES THE PIECEWISE CUBIC SPLINE ORDINATE YY AT THE
C  ABSCISSA XX, WHERE XX IS IN THE CLOSED INTERVAL (X(1),X(M)).
C  NOTE: IF COMPUTING OVER EQUAL INTERVALS, USE THE SUBR 'CUBIC'
C  WHICH REQUIRES ONLY ONE CALL.
C
      DIMENSION X(1),Y(1),A(1),B(1),C(1)
      IF(XX.LT.X(1).OR.XX.GT.X(M)) GO TO 9
      M1=M-1
      DO 1 I=1,M1
      J=I
      IF(XX.LE.X(I+1)) GO TO 2
    1 CONTINUE
    9 WRITE(6,60) XX,X(1),X(M)
   60 FORMAT('0ERROR IN SPOINT CALL--XX=',E16.8,' NOT IN CLOSED INTERVAL
     * (',E16.8,',',E16.8,')')
      RETURN
    2 Z=XX-X(J)
      YY=Y(J)+((C(J)*Z+B(J))*Z+A(J))*Z
      RETURN
      END
      SUBROUTINE WARN(MSG,ISKIP,IUNIT1,IUNIT2,*)
C
C  GENERAL WARNING MESSAGE OUTPUT AND RETURN 1 ON VAX-11/780
C
C  MSG*(*) = VARIABLE-LENGTH 'MESSAGE'
C  ISKIP = 0 FOR NO BLANK LINE BEFORE OUTPUT TO IUNIT1 & IUNIT2
C        > 0 FOR ONE BLANK LINE BEFORE.
C  IUNIT1 = 0 TO SUPPRESS OUTPUT ON IUNIT1 (>0 TO WRITE ON IUNIT1).
C  IUNIT2 = 0 TO SUPPRESS OUTPUT ON IUNIT2 (>0 TO WRITE ON IUNIT2).
C
C  MESSAGES ARE WRITTEN IN THE FORM:
C
C  {WARN}: _MSG_HERE_
C
      CHARACTER*(*) MSG
      I=LEN(MSG)
      DO 1 J=1,2
        IF(J.EQ.1) THEN
       JUNIT=IUNIT1
          ELSE
       JUNIT=IUNIT2
        ENDIF
        IF(JUNIT.GT.0) THEN
          IF(ISKIP.EQ.0) THEN
            WRITE(JUNIT,2) MSG
            ELSE
       WRITE(JUNIT,3) MSG
          ENDIF
      ENDIF
1     CONTINUE
      RETURN 1
2     FORMAT(1X,'{WARN}: ',A<I>)
3     FORMAT(/1X,'{WARN}: ',A<I>)
```

```
      END
      COMPLEX FUNCTION ZHANKS(N,B,FUN,TOL,NF,NEW)
C (VAX-11/780 VERSION FORTRAN-77 (X3.9-1978); SEE NOTE(2) BELOW.)
C=======================================================================
C  COMPLEX HANKEL TRANSFORMS OF ORDER 0 OR 1 FOR RELATED (SAVED) KERNELS
C  AND FIXED TRANSFORM ARGUMENT B.GT.0.
C
C--REF: ANDERSON, W.L., 1979, GEOPHYSICS, VOL. 44, NO. 7, P. 1287-1305.
C
C--SUBPROGRAM ZHANKS EVALUATES THE INTEGRAL FROM 0 TO INFINITY OF
C  FUN(G)*JN(G*B)*DG, DEFINED AS THE COMPLEX HANKEL TRANSFORM OF
C  ORDER N (=0 OR 1) AND TRANSFORM ARGUMENT B.GT.0.  THE METHOD IS BY
C  ADAPTIVE DIGITAL FILTERING OF THE COMPLEX KERNEL FUNCTION FUN,
C  USING DIRECT AND/OR PREVIOUSLY SAVED KERNEL FUNCTION VALUES.
C
C--PARAMETERS (ALL INPUT, EXCEPT NF)
C
C      N      = ORDER (=0 OR 1) OF THE HANKEL TRANSFORM TO BE EVALUATED.
C      B      = REAL TRANSFORM ARGUMENT B.GT.0.0 OF THE HANKEL TRANSFORM.
C               IF NEW=0, B IS ASSUMED EQUAL TO THE LAST B USED WHEN NEW=1
C               (SEE PARAMETER NEW AND SUBPROGRAM USAGE BELOW).
C      FUN(G)= EXTERNAL DECLARED COMPLEX FUNCTION NAME (USER SUPPLIED)
C               OF A REAL ARGUMENT G.GT.0. THIS REFERENCE MUST BE SUPPLIED
C               EVEN WHEN NEW=0, SINCE THE ADAPTIVE CONVOLUTION
C               MAY NEED SOME DIRECT FUNCTION CALLS (E.G. IF TOL REDUCED).
C               IF PARAMETERS OTHER THAN G ARE REQUIRED IN FUN, USE COMMON
C               IN THE CALLING PROGRAM AND IN SUBPROGRAM FUN.  BOTH
C               REAL AND IMAGINARY PARTS OF THE COMPLEX FUNCTION FUN(G)
C               MUST BE CONTINUOUS BOUNDED FUNCTIONS FOR G.GT.0.0. FOR A
C               REAL FUNCTION F1(G), FUN=CMPLX(F1(G),0.0) MAY BE USED.
C               TWO INDEPENDENT REAL-FUNCTIONS F1(G),F2(G) MAY BE
C               INTEGRATED IN PARALLEL BY WRITING FUN=CMPLX(F1(G),F2(G)).
C      TOL    = REQUESTED REAL TRUNCATION TOLERANCE ACCEPTED AT THE FILTER
C               TAILS FOR ADAPTIVE FILTERING.  A TRUNCATION CRITERION IS
C               DEFINED DURING CONVOLUTION IN A FIXED ABSCISSA RANGE AS
C               THE MAX. ABSOLUTE CONVOLVED PRODUCT TIMES TOL.  TYPICALLY,
C               TOL.LE.0.00001 WOULD GIVE ABOUT .01 PER CENT ACCURACY
C               FOR WELL-BEHAVED KERNELS AND MODERATE VALUES OF B.  FOR
C               VERY LARGE OR SMALL B, A VERY SMALL TOL SHOULD BE USED.
C               IN GENERAL, DECREASING THE TOLERANCE WOULD PRODUCE HIGHER
C               ACCURACY IN THE CONVOLUTION SINCE MORE FILTER WEIGHTS ARE
C               USED (UNLESS EXPONENT UNDERFLOWS OCCUR IN THE KERNEL
C               EVALUATION -- SEE NOTE (1) BELOW).
C               FOR MAXIMUM ACCURACY POSSIBLE, TOL=0.0 MAY BE USED.
C      NF     = TOTAL NUMBER OF DIRECT FUN CALLS USED DURING CONVOLUTION
C               FOR ANY VALUE OF NEW (NF IS AN OUTPUT PARAMETER).
C               NF IS IN THE RANGE 21.LE.NF.LE.283 WHEN NEW=1.  USUALLY,
C               NF IS MUCH LESS THAN 283 (OR 0) WHEN NEW=0.
C      NEW   =1 IS REQUIRED FOR THE VERY FIRST CALL TO ZHANKS, OR IF
C               FORCING DIRECT FUNCTION FUN(G) CALLS, E.G., IF USING
C               ZHANKS FOR UNRELATED KERNELS.
C               NEW=1 INITIALIZES COMMON/SAVE/FSAVE(283),GSAVE(283),NSAVE
C               FOR NSAVE COMPLEX KERNEL VALUES IN FSAVE AND CORRESPONDING
C               REAL ARGUMENTS IN GSAVE FOR THE GIVEN PARAMETER B.
C      NEW   =0 TO USE RELATED KERNELS (MODIFIED BY USER) CURRENTLY STORED
```

```
C               IN COMMON/SAVE/. FUN IS CALLED ONLY IF REQUIRED
C               DURING THE CONVOLUTION.  ADDITIONAL FUNCTION VALUES WHEN
C               NEEDED ARE AUTOMATICALLY ADDED TO THE COMMON/SAVE/ BLOCK.
C
C       ******* NOTE THAT IT IS THE USERS RESPONSIBILITY TO MODIFY THE
C               COMMON FSAVE() VALUES FOR NEW=0 CALLS, EXTERNALLY IN
C               THE USERS CALLING PROGRAM (SEE SUBPROGRAM USAGE BELOW).
C
C=======================================================================
C--SUBPROGRAM USAGE-- ZHANKS IS CALLED AS FOLLOWS
C       ...
C       COMPLEX Z1,Z2,ZHANKS,FSAVE
C       COMMON/SAVE/FSAVE(283),GSAVE(283),NSAVE
C       EXTERNAL ZF1,ZF2
C       ...
C       Z1=ZHANKS(N1,B,ZF1,TOL,NF1,1)
C       DO 1 I=1,NSAVE
C    C--MODIFY FSAVE IN COMMON/SAVE/ TO OBTAIN RELATED ZF2 FROM ZF1.
C    C--E.G. FSAVE(I)=GSAVE(I)*FSAVE(I) -- FOR RELATION ZF2(G)=G*ZF1(G)
C     1 CONTINUE
C       Z2=ZHANKS(N2,B,ZF2,TOL,NF2,0)
C       ...
C       END
C       COMPLEX FUNCTION ZF1(G)
C       ...USER SUPPLIED CODE FOR DIRECT EVALUATION OF ZF1(G), G.GT.0.
C       END
C       COMPLEX FUNCTION ZF2(G)
C       ...USER SUPPLIED CODE FOR DIRECT EVALUATION OF ZF2(G), G.GT.0.
C       END
C=======================================================================
C--NOTES
C          (1).  EXP-UNDERFLOW MAY OCCUR IN EXECUTING THIS SUBPROGRAM.
C                THIS IS OK PROVIDED THE MACHINE SYSTEM CONDITIONALLY SETS
C                EXP-UNDERFLOW TO 0.0.
C          (2).  ANSI FORTRAN (AMERICAN STANDARD X3.9-1966) IS USED, EXCEPT
C                DATA STATEMENTS MAY NEED TO BE CHANGED FOR SOME COMPILERS.
C                TO CONVERT ZHANKS TO THE NEW AMERICAN STANDARD FORTRAN
C                (X3.9-1978), ADD THE FOLLOWING DECLARATION TO THIS ROUTINE
C                SAVE Y1,ISAVE
C          (3).  THE FILTER ABSCISSA CORRESPONDING TO EACH FILTER WEIGHT
C                IS GENERATED IN DOUBLE-PRECISION (TO REDUCE ROUND-OFF),
C                BUT IS USED IN SINGLE-PRECISION IN FUNCTION FUN.
C          (4).  NO CHECKS ARE MADE ON CALLING PARAMETERS (TO SAVE TIME),
C                HENCE UNPREDICTABLE RESULTS COULD OCCUR IF ZHANKS
C                IS CALLED INCORRECTLY (OR IF FUN OR COMMON IS IN ERROR).
C=======================================================================
C
        SAVE Y1,ISAVE
        COMPLEX FUN,C,CMAX,FSAVE
        COMMON/SAVE/FSAVE(283),GSAVE(283),NSAVE
        DOUBLE PRECISION E,ER,Y1,Y
        DIMENSION T(2),TMAX(2)
        DIMENSION WT0(283),WA0(76),WB0(76),WC0(76),WD0(55),
       * WT1(283),WA1(76),WB1(76),WC1(76),WD1(55)
        EQUIVALENCE (WT0(1),WA0(1)),(WT0(77),WB0(1)),(WT0(153),WC0(1)),
     * (WT0(229),WD0(1)),(WT1(1),WA1(1)),(WT1(77),WB1(1)),
     * (WT1(153),WC1(1)),(WT1(229),WD1(1))
      EQUIVALENCE (C,T(1)),(CMAX,TMAX(1))
C-----E=DEXP(.2D0), ER=1.0D0/E
      DATA E/1.221402758160169834 D0/,ER/.818730753077981859 D0/
C--J0-TRANSFORM FILTER WEIGHT ARRAYS (EQUIVALENT TO WT0 ARRAY)
      DATA WA0/
     * 2.1969101E-11, 4.1201161E-09,-6.1322980E-09, 7.2479291E-09,
     *-7.9821627E-09, 8.5778983E-09,-9.1157294E-09, 9.6615250E-09,
     *-1.0207546E-08, 1.0796633E-08,-1.1393033E-08, 1.2049873E-08,
     *-1.2708789E-08, 1.3446466E-08,-1.4174300E-08, 1.5005577E-08,
     *-1.5807160E-08, 1.6747136E-08,-1.7625961E-08, 1.8693427E-08,
     *-1.9650840E-08, 2.0869789E-08,-2.1903555E-08, 2.3305308E-08,
     *-2.4407377E-08, 2.6033678E-08,-2.7186773E-08, 2.9094334E-08,
     *-3.0266804E-08, 3.2534013E-08,-3.3672072E-08, 3.6408936E-08,
     *-3.7425022E-08, 4.0787921E-08,-4.1543242E-08, 4.5756842E-08,
     *-4.6035233E-08, 5.1425075E-08,-5.0893896E-08, 5.7934897E-08,
     *-5.6086570E-08, 6.5475248E-08,-6.1539913E-08, 7.4301996E-08,
     *-6.7117043E-08, 8.4767837E-08,-7.2583120E-08, 9.7366568E-08,
     *-7.7553611E-08, 1.1279873E-07,-8.1416723E-08, 1.3206914E-07,
     *-8.3217217E-08, 1.5663185E-07,-8.1482581E-08, 1.8860593E-07,
     *-7.3963141E-08, 2.3109673E-07,-5.7243707E-08, 2.8867452E-07,
     *-2.6163525E-08, 3.6808773E-07, 2.7049871E-08, 4.7932617E-07,
     * 1.1407365E-07, 6.3720626E-07, 2.5241961E-07, 8.6373487E-07,
     * 4.6831433E-07, 1.1916346E-06, 8.0099716E-07, 1.6696015E-06,
     * 1.3091334E-06, 2.3701475E-06, 2.0803829E-06, 3.4012978E-06/
      DATA WB0/
     * 3.2456774E-06, 4.9240402E-06, 5.0005198E-06, 7.1783540E-06,
     * 7.6367633E-06, 1.0522038E-05, 1.1590021E-05, 1.5488635E-05,
     * 1.7510398E-05, 2.2873836E-05, 2.6368006E-05, 3.3864387E-05,
     * 3.9610390E-05, 5.0230379E-05, 5.9397373E-05, 7.4612122E-05,
     * 8.8951409E-05, 1.1094809E-04, 1.3308026E-04, 1.6511335E-04,
     * 1.9895671E-04, 2.4587195E-04, 2.9728181E-04, 3.6629770E-04,
     * 4.4402013E-04, 5.4589361E-04, 6.6298832E-04, 8.1375348E-04,
     * 9.8971624E-04, 1.2132772E-03, 1.4772052E-03, 1.8092022E-03,
     * 2.2045122E-03, 2.6980811E-03, 3.2895354E-03, 4.0238764E-03,
     * 4.9080203E-03, 6.0010999E-03, 7.3216878E-03, 8.9489225E-03,
     * 1.0919448E-02, 1.3340696E-02, 1.6276399E-02, 1.9873311E-02,
     * 2.4233627E-02, 2.9555699E-02, 3.5990069E-02, 4.3791529E-02,
     * 5.3150319E-02, 6.4341372E-02, 7.7506720E-02, 9.2749987E-02,
     * 1.0980561E-01, 1.2791555E-01, 1.4525830E-01, 1.5820085E-01,
     * 1.6058576E-01, 1.4196085E-01, 8.9781222E-02,-1.0238278E-02,
     *-1.5083434E-01,-2.9059573E-01,-2.9105437E-01,-3.7973244E-02,
     * 3.8273717E-01, 2.2014118E-01,-4.7342635E-01, 1.9331133E-01,
     * 5.3839527E-02,-1.1909845E-01, 9.9317051E-02,-6.6152628E-02,
     * 4.0703241E-02,-2.4358316E-02, 1.4476533E-02,-8.6198067E-03/
      DATA WC0/
     * 5.1597053E-03,-3.1074602E-03, 1.8822342E-03,-1.1456545E-03,
     * 7.0004347E-04,-4.2904226E-04, 2.6354444E-04,-1.6215439E-04,
     * 9.9891279E-05,-6.1589037E-05, 3.7996921E-05,-2.3452250E-05,
     * 1.4479572E-05,-8.9417427E-06, 5.5227518E-06,-3.4114252E-06,
     * 2.1074101E-06,-1.3019229E-06, 8.0433617E-07,-4.9693681E-07,
     * 3.0702417E-07,-1.8969219E-07, 1.1720069E-07,-7.2412496E-08,
     * 4.4740283E-08,-2.7643004E-08, 1.7079403E-08,-1.0552634E-08,
     * 6.5200311E-09,-4.0284597E-09, 2.4890232E-09,-1.5378695E-09,
     * 9.5019040E-10,-5.8708696E-10, 3.6273937E-10,-2.2412348E-10,      00025640
     * 1.3847792E-10,-8.5560821E-11, 5.2865474E-11,-3.2664392E-11,      00025650
     * 2.0182948E-11,-1.2470979E-11, 7.7057678E-12,-4.7611713E-12,      00025660
     * 2.9415274E-12,-1.8170081E-12, 1.1221034E-12,-6.9271067E-13,      00025670
     * 4.2739744E-13,-2.6344388E-13, 1.6197105E-13,-9.9147443E-14,      00025680
     * 6.0487998E-14,-3.6973097E-14, 2.2817964E-14,-1.4315547E-14,      00025690
     * 9.1574735E-15,-5.9567236E-15, 3.9209969E-15,-2.5911739E-15,      00025700
     * 1.6406939E-15,-8.8248590E-16, 3.0195409E-16, 2.2622634E-17,      00025710
     *-8.0942556E-17,-3.7172363E-17, 1.9299542E-16,-3.3388160E-16,      00025720
     * 4.6174116E-16,-5.8627358E-16, 7.2227767E-16,-8.7972941E-16,      00025730
     * 1.0211793E-15,-1.0940039E-15, 1.0789555E-15,-9.7089714E-16/      00025740
      DATA WD0/                                                         00025750
     * 7.4110927E-16,-4.1700094E-16, 8.5977184E-17, 1.3396469E-16,      00025760
     *-1.7838410E-16, 4.8975421E-17, 1.9398153E-16,-5.0046989E-16,      00025770
     * 8.3280985E-16,-1.1544640E-15, 1.4401527E-15,-1.6637066E-15,      00025780
     * 1.7777129E-15,-1.7322187E-15, 1.5247247E-15,-1.1771155E-15,      00025790
     * 6.9747910E-16,-1.2088956E-16,-4.8382957E-16, 1.0408292E-15,      00025800
     *-1.5220450E-15, 1.9541597E-15,-2.4107448E-15, 2.9241438E-15,      00025810
     *-3.5176475E-15, 4.2276125E-15,-5.0977851E-15, 6.1428456E-15,      00025820
     *-7.3949962E-15, 8.8597601E-15,-1.0515959E-14, 1.2264584E-14,      00025830
     *-1.3949870E-14, 1.5332490E-14,-1.6146782E-14, 1.6084121E-14,      00025840
     *-1.4962523E-14, 1.2794804E-14,-9.9286701E-15, 6.8825809E-15,      00025850
     *-4.0056107E-15, 1.5965079E-15,-7.2732961E-18,-4.0433218E-16,      00025860
     *-6.5679655E-16, 3.3011866E-15,-7.3545910E-15, 1.2394851E-14,      00025870
     *-1.7947697E-14, 2.3774303E-14,-3.0279168E-14, 3.9252831E-14,      00025880
     *-5.5510504E-14, 9.0505371E-14,-1.7064873E-13/                     00025890
C--END OF J0 FILTER WEIGHTS                                             00025900
C                                                                       00025910
C--J1-TRANSFORM FILTER WEIGHT ARRAYS (EQUIVALENT TO WT1 ARRAY)          00025920
      DATA WA1/                                                         00025930
     *-4.2129715E-16, 5.3667031E-15,-7.1183962E-15, 8.9478500E-15,      00025940
     *-1.0767891E-14, 1.2362265E-14,-1.3371129E-14, 1.3284178E-14,      00025950
     *-1.1714302E-14, 8.4134738E-15,-3.7726725E-15,-1.4263879E-15,      00025960
     * 6.1279163E-15,-9.1102765E-15, 9.9696405E-15,-9.3649955E-15,      00025970
     * 8.6009018E-15,-8.9749846E-15, 1.1153987E-14,-1.4914821E-14,      00025980
     * 1.9314024E-14,-2.3172388E-14, 2.5605477E-14,-2.6217555E-14,      00025990
     * 2.5057768E-14,-2.2485539E-14, 1.9022752E-14,-1.5198084E-14,      00026000
     * 1.1422464E-14,-7.9323958E-15, 4.8421406E-15,-2.1875032E-15,      00026010
     *-3.2177842E-17, 1.8637565E-15,-3.3683643E-15, 4.6132219E-15,      00026020
     *-5.6209538E-15, 6.4192841E-15,-6.8959928E-15, 6.9895792E-15,      00026030
     *-6.5355935E-15, 5.6125163E-15,-4.1453931E-15, 2.6358827E-15,      00026040
     *-9.5104370E-16, 1.4600474E-16, 5.6166519E-16, 8.2899246E-17,      00026050
     * 5.0032100E-16, 4.3752205E-16, 2.1052293E-15,-9.5451973E-16,      00026060
     * 6.4004437E-15,-2.1926177E-15, 1.1651003E-14, 5.8415433E-16,      00026070
     * 1.8044664E-14, 1.0755745E-14, 3.0159022E-14, 3.3506138E-14,      00026080
     * 5.8709354E-14, 8.1475200E-14, 1.2530006E-13, 1.8519112E-13,      00026090
     * 2.7641786E-13, 4.1330823E-13, 6.1506209E-13, 9.1921659E-13,      00026100
     * 1.3698462E-12, 2.0447427E-12, 3.0494477E-12, 4.5501001E-12,      00026110
     * 6.7870250E-12, 1.0126237E-11, 1.5104976E-11, 2.2536053E-11/      00026120
      DATA WB1/                                                         00026130
     * 3.3617368E-11, 5.0153839E-11, 7.4818173E-11, 1.1161804E-10,      00026140
     * 1.6651222E-10, 2.4840923E-10, 3.7058109E-10, 5.5284353E-10,      00026150
     * 8.2474468E-10, 1.2303750E-09, 1.8355034E-09, 2.7382502E-09,      00026160
     * 4.0849867E-09, 6.0940898E-09, 9.0913020E-09, 1.3562651E-08,      00026170
     * 2.0233058E-08, 3.0184244E-08, 4.5029477E-08, 6.7176304E-08,      00026180
     * 1.0021488E-07, 1.4950371E-07, 2.2303208E-07, 3.3272689E-07,
     * 4.9636623E-07, 7.4049804E-07, 1.1046805E-06, 1.6480103E-06,
     * 2.4585014E-06, 3.6677163E-06, 5.4714550E-06, 8.1626422E-06,
     * 1.2176782E-05, 1.8166179E-05, 2.7099223E-05, 4.0428804E-05,
     * 6.0307294E-05, 8.9971508E-05, 1.3420195E-04, 2.0021123E-04,
     * 2.9860417E-04, 4.4545291E-04, 6.6423156E-04, 9.9073275E-04,
     * 1.4767050E-03, 2.2016806E-03, 3.2788147E-03, 4.8837292E-03,
     * 7.2596811E-03, 1.0788355E-02, 1.5973323E-02, 2.3612041E-02,
     * 3.4655327E-02, 5.0608141E-02, 7.2827752E-02, 1.0337889E-01,
     * 1.4207357E-01, 1.8821315E-01, 2.2996815E-01, 2.5088500E-01,
     * 2.0334626E-01, 6.0665451E-02,-2.0275683E-01,-3.5772336E-01,
     *-1.8280529E-01, 4.7014634E-01, 7.2991233E-03,-3.0614594E-01,
     * 2.4781735E-01,-1.1149185E-01, 2.5985386E-02, 1.0850279E-02,
     *-2.2830217E-02, 2.4644647E-02,-2.2895284E-02, 2.0197032E-02/
      DATA WC1/
     *-1.7488968E-02, 1.5057670E-02,-1.2953923E-02, 1.1153254E-02,
     *-9.6138436E-03, 8.2952090E-03,-7.1628361E-03, 6.1882910E-03,
     *-5.3482055E-03, 4.6232056E-03,-3.9970542E-03, 3.4560118E-03,
     *-2.9883670E-03, 2.5840861E-03,-2.2345428E-03, 1.9323046E-03,
     *-1.6709583E-03, 1.4449655E-03,-1.2495408E-03, 1.0805480E-03,
     *-9.3441130E-04, 8.0803899E-04,-6.9875784E-04, 6.0425624E-04,
     *-5.2253532E-04, 4.5186652E-04,-3.9075515E-04, 3.3790861E-04,
     *-2.9220916E-04, 2.5269019E-04,-2.1851585E-04, 1.8896332E-04,
     *-1.6340753E-04, 1.4130796E-04,-1.2219719E-04, 1.0567099E-04,
     *-9.1379828E-05, 7.9021432E-05,-6.8334412E-05, 5.9092726E-05,
     *-5.1100905E-05, 4.4189914E-05,-3.8213580E-05, 3.3045496E-05,
     *-2.8576356E-05, 2.4711631E-05,-2.1369580E-05, 1.8479514E-05,
     *-1.5980307E-05, 1.3819097E-05,-1.1950174E-05, 1.0334008E-05,
     *-8.9364160E-06, 7.7278366E-06,-6.6827083E-06, 5.7789251E-06,
     *-4.9973715E-06, 4.3215167E-06,-3.7370660E-06, 3.2316575E-06,
     *-2.7946015E-06, 2.4166539E-06,-2.0898207E-06, 1.8071890E-06,
     *-1.5627811E-06, 1.3514274E-06,-1.1686576E-06, 1.0106059E-06,
     *-8.7392952E-07, 7.5573750E-07,-6.5353002E-07, 5.6514528E-07,
     *-4.8871388E-07, 4.2261921E-07,-3.6546333E-07, 3.1603732E-07/
      DATA WD1/
     *-2.7329579E-07, 2.3633470E-07,-2.0437231E-07, 1.7673258E-07,
     *-1.5283091E-07, 1.3216174E-07,-1.1428792E-07, 9.8831386E-08,
     *-8.5465227E-08, 7.3906734E-08,-6.3911437E-08, 5.5267923E-08,
     *-4.7793376E-08, 4.1329702E-08,-3.5740189E-08, 3.0906612E-08,
     *-2.6726739E-08, 2.3112160E-08,-1.9986424E-08, 1.7283419E-08,
     *-1.4945974E-08, 1.2924650E-08,-1.1176694E-08, 9.6651347E-09,
     *-8.3580023E-09, 7.2276490E-09,-6.2501673E-09, 5.4048822E-09,
     *-4.6739154E-09, 4.0418061E-09,-3.4951847E-09, 3.0224895E-09,
     *-2.6137226E-09, 2.2602382E-09,-1.9545596E-09, 1.6902214E-09,
     *-1.4616324E-09, 1.2639577E-09,-1.0930164E-09, 9.4519327E-10,
     *-8.1736202E-10, 7.0681930E-10,-6.1122713E-10, 5.2856342E-10,
     *-4.5707937E-10, 3.9526267E-10,-3.4180569E-10, 2.9557785E-10,
     *-2.5560176E-10, 2.2103233E-10,-1.9113891E-10, 1.6528994E-10,
     *-1.4294012E-10, 1.2361991E-10,-8.2740936E-11/
C--END OF J1 FILTER WEIGHTS
C
      NONE=0
      IF(NEW.EQ.0) GO TO 100
      NSAVE=0
C-----INITIALIZE KERNEL ABSCISSA GENERATION FOR GIVEN B
```

```
      Y1=0.735885266147979446OD0/DBLE(B)
  100 ZHANKS=(0.0,0.0)
      CMAX=(0.0,0.0)
      NF=0
      Y=Y1
C-----BEGIN RIGHT-SIDE CONVOLUTION AT WEIGHT 131 (EITHER NEW=1 OR 0)
      ASSIGN 110 TO M
      I=131
      Y=Y*E
      GO TO 200
  110 TMAX(1)=AMAX1(ABS(T(1)),TMAX(1))
      TMAX(2)=AMAX1(ABS(T(2)),TMAX(2))
      I=I+1
      Y=Y*E
      IF(I.LE.149) GO TO 200
      IF(TMAX(1).EQ.0.0.AND.TMAX(2).EQ.0.0) NONE=1
C-----ESTABLISH TRUNCATION CRITERION (CMAX=CMPLX(TMAX(1),TMAX(2))
      CMAX=TOL*CMAX
      ASSIGN 120 TO M
      GO TO 200
C-----CHECK FOR FILTER TRUNCATION AT RIGHT END
  120 IF(ABS(T(1)).LE.TMAX(1).AND.ABS(T(2)).LE.TMAX(2)) GO TO 130
      I=I+1
      Y=Y*E
      IF(I.LE.283) GO TO 200
  130 Y=Y1
C-----CONTINUE WITH LEFT-SIDE CONVOLUTION AT WEIGHT 130
      ASSIGN 140 TO M
      I=130
      GO TO 200
C-----CHECK FOR FILTER TRUNCATION AT LEFT END
  140 IF(ABS(T(1)).LE.TMAX(1).AND.ABS(T(2)).LE.TMAX(2).AND.
     * NONE.EQ.0) GO TO 190
      I=I-1
      Y=Y*ER
      IF(I.GT.0) GO TO 200
C-----RETURN WITH ISAVE=1 PRESET FOR POSSIBLE NEW=0 USE.
  190 ISAVE=1
C-----NORMALIZE BY B TO ACCOUNT FOR INTEGRATION RANGE CHANGE
      ZHANKS=ZHANKS/B
      RETURN
C-----SAVE/RETRIEVE PSEUDO-SUBROUTINE (CALL FUN ONLY WHEN NECESSARY)
  200 G=SNGL(Y)
      IF(NEW) 300,210,300
  210 IF(ISAVE.GT.NSAVE) GO TO 300
      ISAVE0=ISAVE
  220 IF(G.EQ.GSAVE(ISAVE)) GO TO 240
      ISAVE=ISAVE+1
      IF(ISAVE.LE.NSAVE) GO TO 220
      ISAVE=ISAVE0
C-----G NOT IN COMMON/SAVE/----- EVALUATE FUN.
      GO TO 300
C-----G FOUND IN COMMON/SAVE/----- USE FSAVE AS GIVEN.
  240 C=FSAVE(ISAVE)
      ISAVE=ISAVE+1
```

```
C-----SWITCH ON ORDER N
  250 IF(N) 270,260,270
  260 C=C*WT0(I)
      GO TO 280
  270 C=C*WT1(I)
  280 ZHANKS=ZHANKS+C
      GO TO M,(110,120,140)
C-----DIRECT FUN EVALUATION (AND ADD TO END OF COMMON/SAVE/)
  300 NSAVE=NSAVE+1
      C=FUN(G)
      NF=NF+1
      FSAVE(NSAVE)=C
      GSAVE(NSAVE)=G
      GO TO 250
      END
      REAL FUNCTION ASINH(X)
C--INVERSE HYPERBOLIC SIN FUNCTION
C
      REAL*8 X2
      X2=X
      ASINH=DLOG(X2+DSQRT(X2*X2+1.0D0))
      RETURN
      END
      FUNCTION ERF(X)
C
C  ERF COMPUTES THE ERROR FUNCTION TO ABOUT 7-PLACES.
C  SEE MATH. OF COMP., V.22,N.101,JAN,1968.
C   ALSO, SEE ERFINV(X).
C
      DIMENSION A1(19),A2(19)
      DATA A1/.70322500,.33050152,.20133975,.10863025,
     1 .46775523E-1,.15398573E-1,.38015077E-2,.69718379E-3,
     2 .94490927E-4,.94328117E-5,.69192752E-6,.37225234E-7,
     3 .14666061E-8,.42261614E-10,.88978652E-12,.13676044E-13,
     4 .15334234E-15,.12536751E-17,.74517E-20/
      DATA A2/.24725517,.14422723,.86989455E-1,.43977338E-1,
     1 .17243963E-1,.50790696E-2,.11086065E-2,.17822802E-3,
     2 .21040458E-4,.18206632E-5,.11533099E-6,.53427503E-8,
     3 .18084859E-9,.44696823E-11,.80606884E-13,.10601364E-14,
     4 .10164928E-16,.710005E-19,0.0/
      IF(X.EQ.0.0) THEN
        ERF=0.0
        RETURN
      ENDIF
        B=2.*X/5.
        S=SIN(B)
        C=COS(B)
        C2=C+C
        ALP=C2*C-1.
        SUM=0.0
        DO 10 N=1,19
          SUM=SUM+(A1(N)+C2*A2(N))*ALP**(N-1)
10      CONTINUE
        ERF=B/3.1415927+S*SUM
        RETURN
      END
      FUNCTION ERFINV(Y)
C
C  ERFINV COMPUTES THE INVERSE ERROR FUNCTION TO ABOUT 7-PLACES.
C  SEE MATH. OF COMP., V.22,N.101,JAN,1968.
C  ALSO, SEE ERF(X).
C
      CHARACTER*16 XX
      DIMENSION T3(1:38),T4(0:26),T5(0:37),T6(0:25)
      DATA T3/.12046752,.16078199E-1,.26867044E-2,.49963473E-3,
     1 .98898219E-4,.20391813E-4,.43272716E-5,.93808141E-6,
     2 .20673472E-6,.46159699E-7,.10416680E-7,.23715100E-8,
     3 .54392841E-9,.12554899E-9,.29138180E-10,.67949422E-11,
     4 .15912343E-11,.37402505E-12,.88208776E-13,.20865090E-13,
     5 .49488041E-14,.11766395E-14,.28038557E-15,.66950664E-16,
     6 .16016550E-16,.38382583E-17,.9212851E-18,.2214615E-18,
     7 .533091E-19,.128488E-19,.31006E-20,.7491E-21,.1812E-21,
     8 .439E-22,.106E-22,.26E-23,.6E-24,.2E-24/
      DATA T4/.91215880,-.16266282E-1,.43355647E-3,.21443857E-3,
     1 .26257511E-5,-.30210911E-5,-.12406061E-7,.62406609E-7,
     2 -.54012479E-9,-.14232079E-8,.34384028E-10,.33584870E-10,
     3 -.14584289E-11,-.81021743E-12,.52532409E-13,.19711541E-13,
     4 -.17494334E-14,-.48005966E-15,.55730299E-16,.11632605E-16,
     5 -.17262489E-17,-.2784973E-18,.524481E-19,.65270E-20,
     6 -.15707E-20,-.1475E-21,.450E-22/
      DATA T5/.95667971,-.23107004E-1,-.43742361E-2,-.57650342E-3,
     1 -.10961022E-4,.25108547E-4,.10562336E-4,.27544123E-5,
     2 .43248450E-6,-.20530336E-7,-.43891537E-7,-.17684010E-7,
     3 -.39912890E-8,-.18693241E-9,.27292274E-9,.13281721E-9,
     4 .31834248E-10,.16700608E-11,-.20364650E-11,-.96484681E-12,
     5 -.21956727E-12,-.95689813E-14,.13703257E-13,.62538505E-14,
     6 .14584615E-14,.10781240E-15,-.70922999E-16,-.39141178E-16,
     7 -.11165921E-16,-.15770366E-17,.2853149E-18,.2716662E-18,
     8 .957770E-19,.176835E-19,-.9828E-21,-.20464E-20,-.802E-21,
     9 -.1650E-21/
      DATA T6/.98857506,.10857705E-1,-.17511651E-2,.21196993E-4,
     1 .15664871E-4,-.51904169E-5,-.37135790E-7,.12174309E-8,
     2 -.17681155E-9,-.11937218E-10,.38025054E-12,-.66018832E-13,
     3 -.87917055E-14,-.35068693E-15,-.69722150E-16,-.10956794E-16,
     4 -.11536390E-17,-.1326235E-18,-.263938E-19,.5341E-21,
     5 -.2261E-20,.9552E-21,-.525E-21,.2487E-21,-.1134E-21,.42E-22/
      X=Y
      X1=ABS(X)
      IF(X1.GE.1.0) THEN
        ENCODE(16,1,XX) X1
1       FORMAT(E16.8)
        IF(X1.GT.1.000001)CALL ERRMSG('ABS(X)='//XX//
     1   ' >1.000001 IN [ERFINV]',0,6,0)
        CALL WARN('ABS(X)='//XX//
     2 ' >=1.0 IN [ERFINV]; X=0.9999998*SIGN(1.,X) USED.',0,6,0,*2)
2       X=0.9999998*SIGN(1.,X)
      ENDIF
      X1=1.-X
      IF(X.GE.0.8.AND.X.LE.0.9975) THEN
        BETA=SQRT(-ALOG(1.-X*X))
```

```
          R=0.0
          DO 10 N=0,26
10          R=R+T4(N)*TCHEB(N,-1.54801304*BETA+2.5654901)
          ERFINV=BETA*R
        ELSE IF(X1.GE.5E-16.AND.X1.LE.25E-4) THEN
          BETA=SQRT(-ALOG(1.-X*X))
          R=0.0
          DO 20 N=0,37
20          R=R+T5(N)*TCHEB(N,-.55945763*BETA+2.2879157)
          ERFINV=BETA*R
        ELSE IF(X1.LT.5E-16) THEN
          BETA=SQRT(-ALOG(1.-X*X))
          SBETA=SQRT(BETA)
          R=0.0
          DO 30 N=0,25
30          R=R+T6(N)*TCHEB(N,-9.1999924/SBETA+2.7949908)
          ERFINV=BETA*R
        ELSE
          R=0.0
          A=X*X/.32-1.
          DO 40 N=1,38
40          R=R+T3(N)*TCHEB(N,A)
          ERFINV=X*(.99288538+R)
        ENDIF
        RETURN
        END
        INTEGER FUNCTION LOC(I,J)
C--GETS ACTUAL ADDR OF A(I,J)=A(J,I) SYMMETRIC MATRIX
C  STORED AS THE VECTOR A(LOC(I,J)) OF N*(N+1)/2 ELEMENTS--
C  WHERE ANY I,J.LE.N MAY BE USED (N NOT EXPLICITLY NEEDED)...
C
        IF(I-J) 10,20,20
   10 LOC=I+(J*J-J)/2
        RETURN
   20 LOC=J+(I*I-I)/2
        RETURN
        END
        SUBROUTINE NL2SOL(N, P, X, CALCR, CALCJ, IV, V, UIPARM, URPARM,
     1          UFPARM)
C** VAX-11/780 VERSION (12/18/80) MODIFIED BY
C** W.L.ANDERSON, U.S.GEOLOGICAL SURVEY, DENVER, COLORADO.
```

$$$$$ Because of the length of NL2SOL and related subprograms, the rest of the listing has been suppressed; however, the complete code is available on the distributed tape.